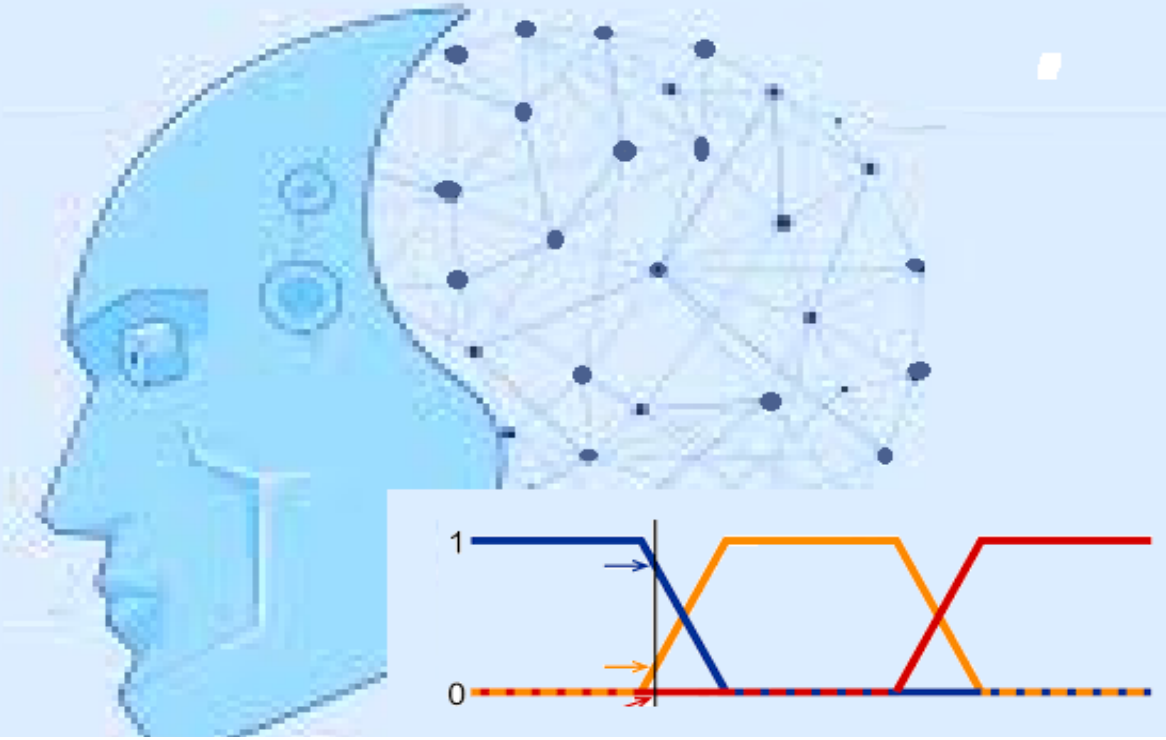
Soft Computing



Module – 1 (Introduction to Soft Computing & Artificial Neural Network)

Introduction to Soft Computing. Difference between Hard Computing & Soft Computing. Applications of Soft Computing. Artificial Neurons Vs Biological Neurons. Basic models of artificial neural networks – Connections, Learning, Activation Functions. McCulloch and Pitts Neuron. Hebb network.

COMPUTING

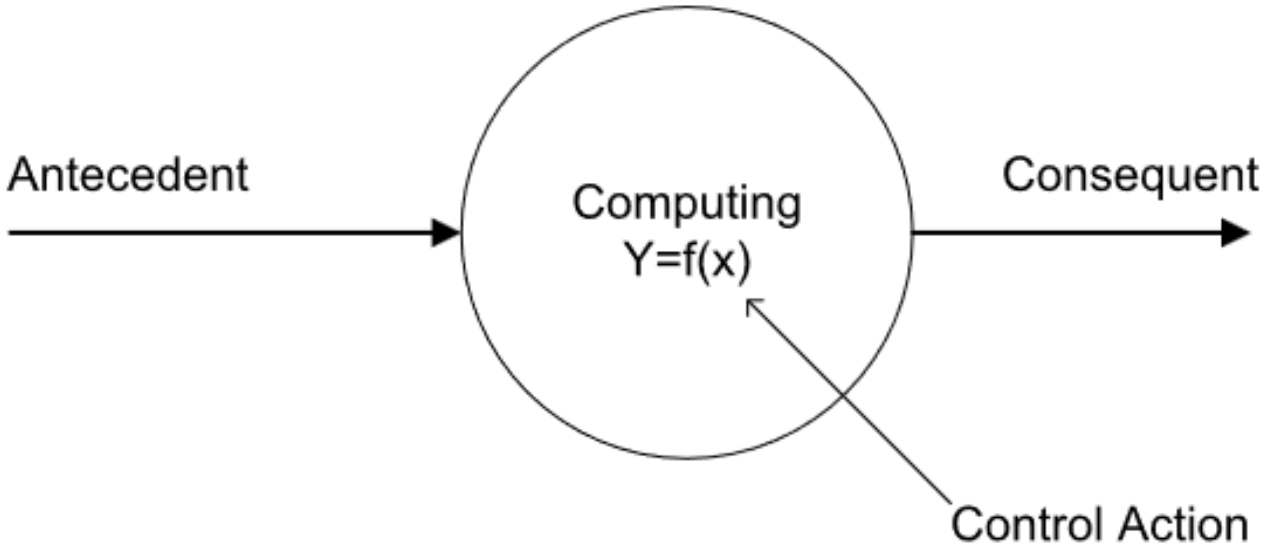


Figure: Basic of computing

- y = f (x), f is a mapping function

f is also called a formal method or an algorithm to solve a problem.

COMPUTING

- Two major problem solving
  - Hard Computing
  - Soft Computing

Hard Computing

- Precise result is guaranteed
- Control action is unambiguous
- Precise model whose accurate solution is achieved
- Control action is formally defined (i.e. with mathematical model)
- Traditional Computing

Example:

- Solving numerical problems (e.g. Roots of polynomials, Integration etc.)
- Searching and sorting techniques
- Solving ”Computational Geometry” problems (e.g. Shortest tour in Graph theory, Finding closest pair of points etc.)

Soft Computing

- Is an emerging approach to computing which parallel the remarkable ability of the human mind to reason and learn in a environment of uncertainly and imprecision

Some of its principle components

Neural Network(NN)

Fuzzy Logic (FL)

Genetic Algorithm(GA)

These form the core of Soft computing

What is SC?

The term soft computing was proposed by the inventor of fuzzy logic, Lotfi A. Zadeh. He describes it as follows [Zadeh, 1994]:

"Soft computing is a collection of methodologies that aim to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost.

Its principal constituents are fuzzy logic, neurocomputing, and probabilistic reasoning.
Soft computing is likely to play an increasingly important role in many application areas, including software engineering.

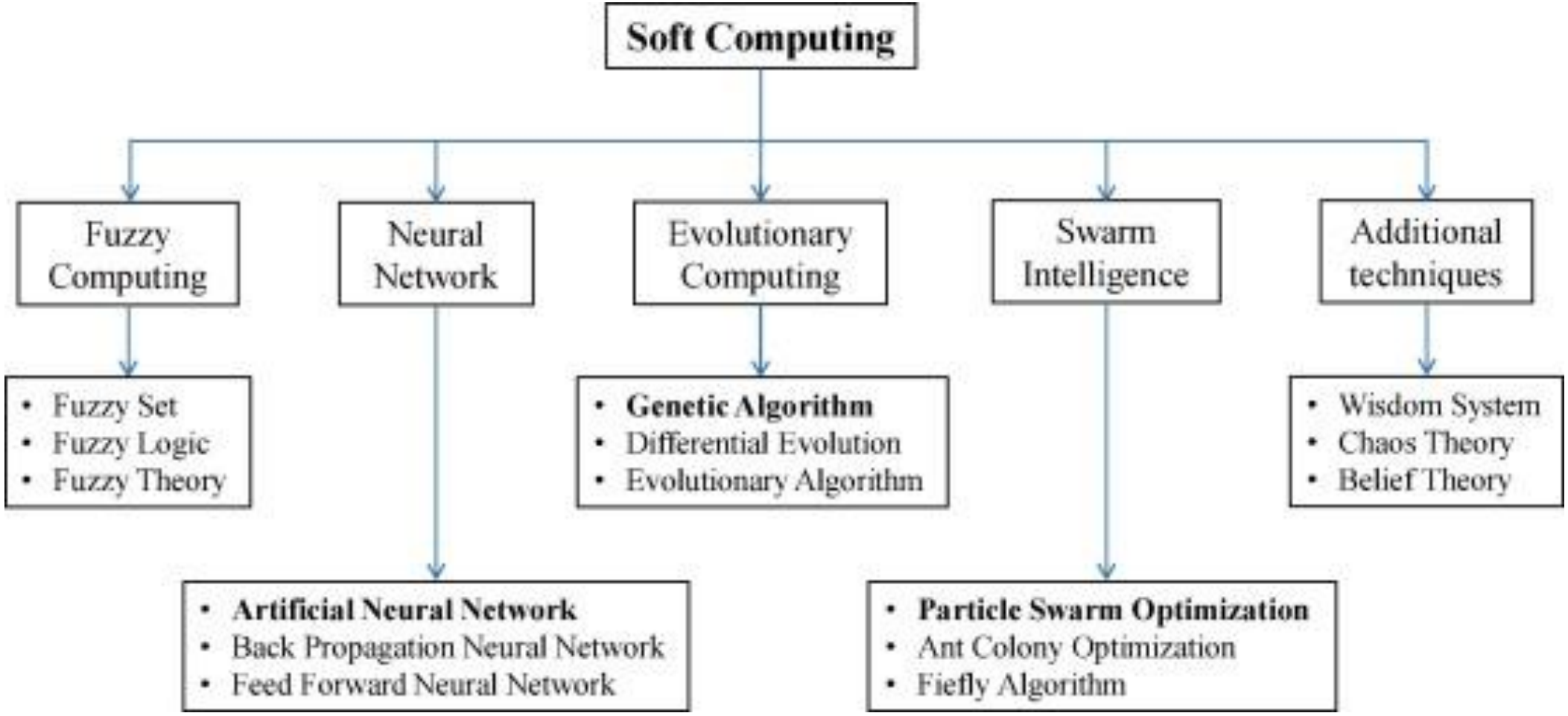
The role model for soft computing is the human mind."

[Zadeh, 1994]

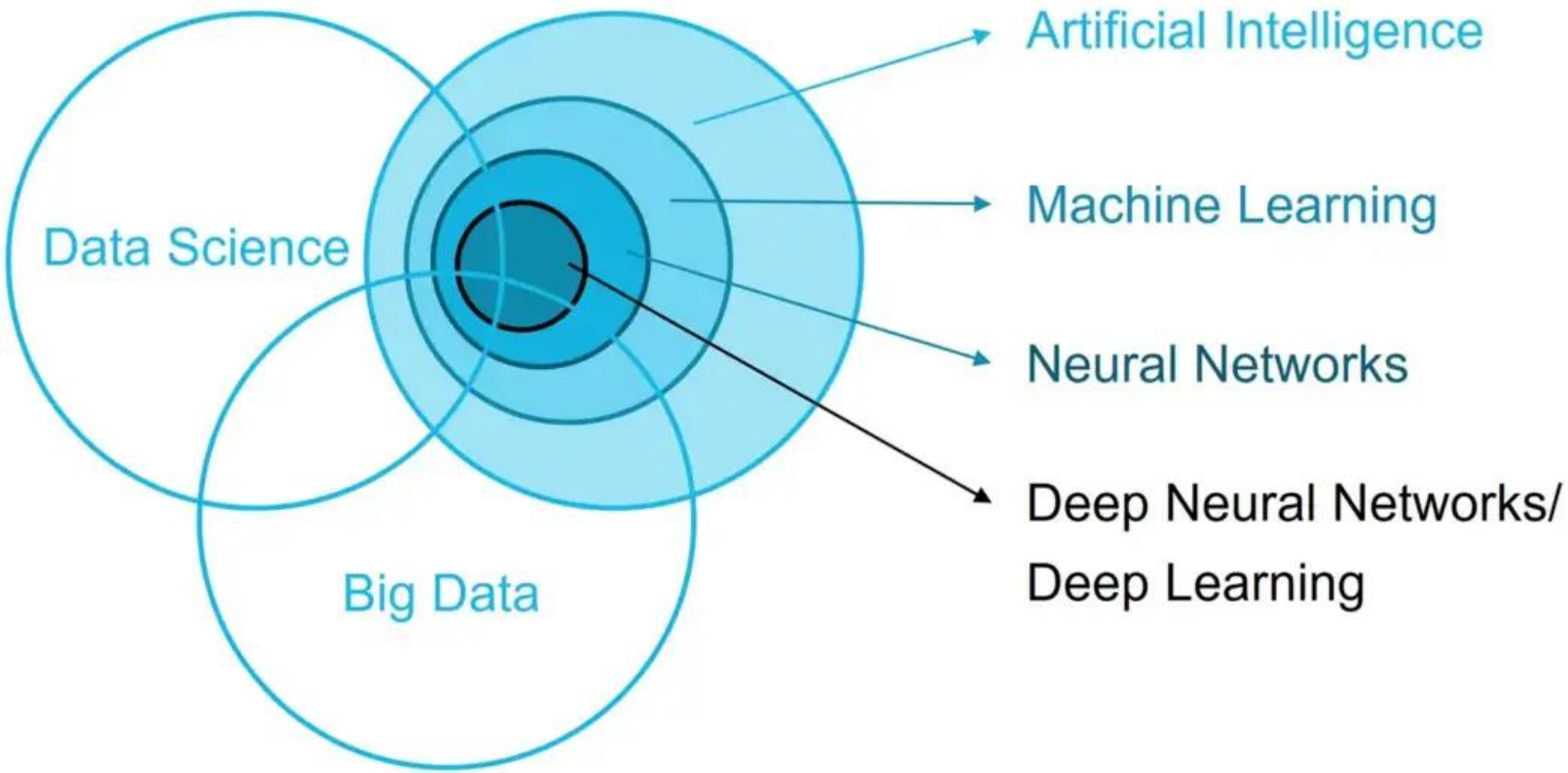
Soft Computing

- “The essence of soft computing is that unlike the traditional, hard computing, soft computing is aimed at an accommodation with the pervasive imprecision of the real world. Thus, the guiding principle of soft computing is to exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, low solution cost, and better rapport with reality”
- - Lotfi Zadeh

Soft Computing



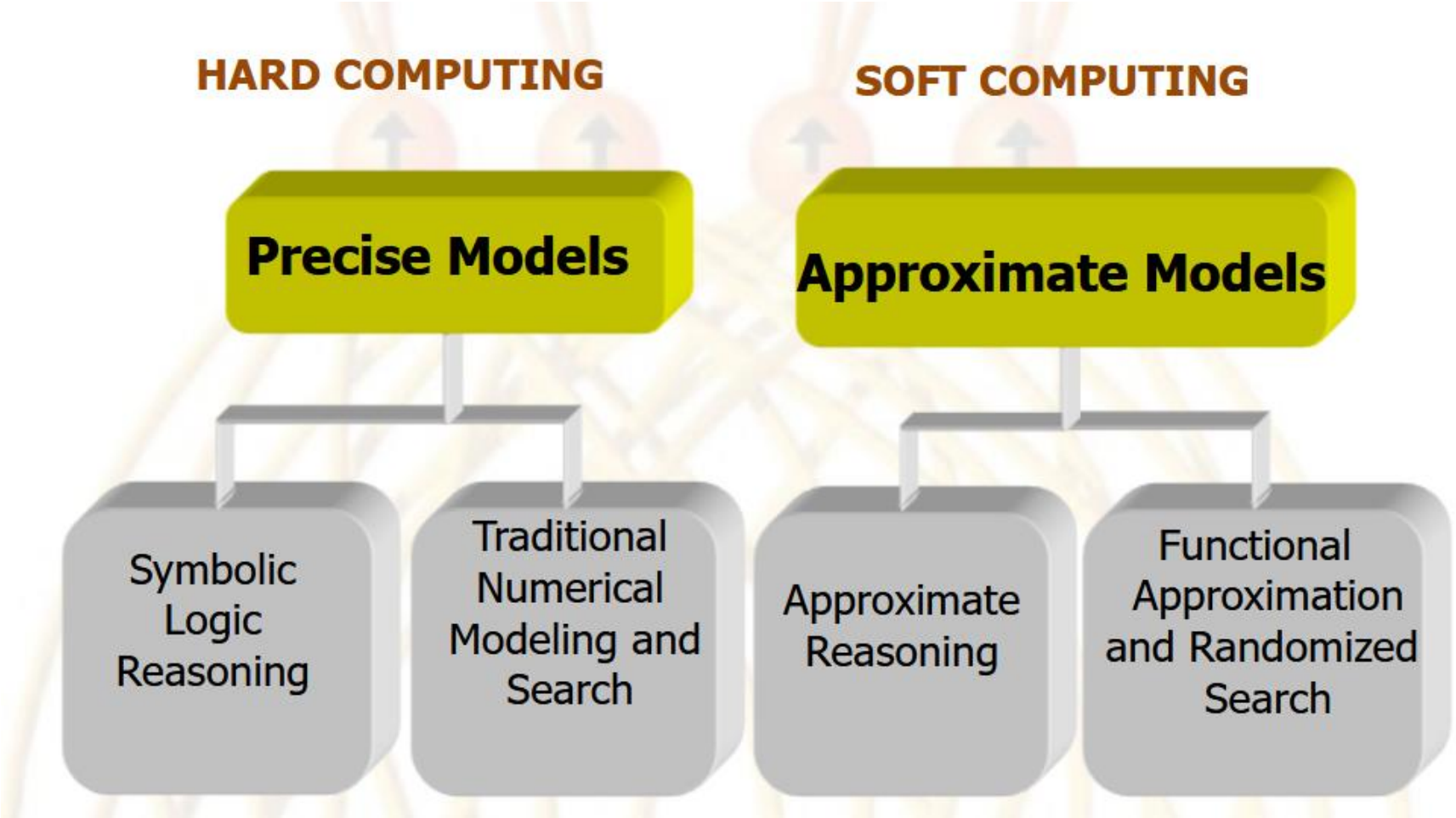
Soft Computing Approaches



Characteristics of Soft Computing

- It does not require any mathematical modeling of problem solving
- It may not yield the precise solution
- Algorithms are adaptive (i.e. it can adjust to the change of dynamic environment)
- Use some biological inspired methodologies such as genetics, evolution, Ant’s behaviors, particles swarming, human nervous systems etc.

Problem Solving Technologies



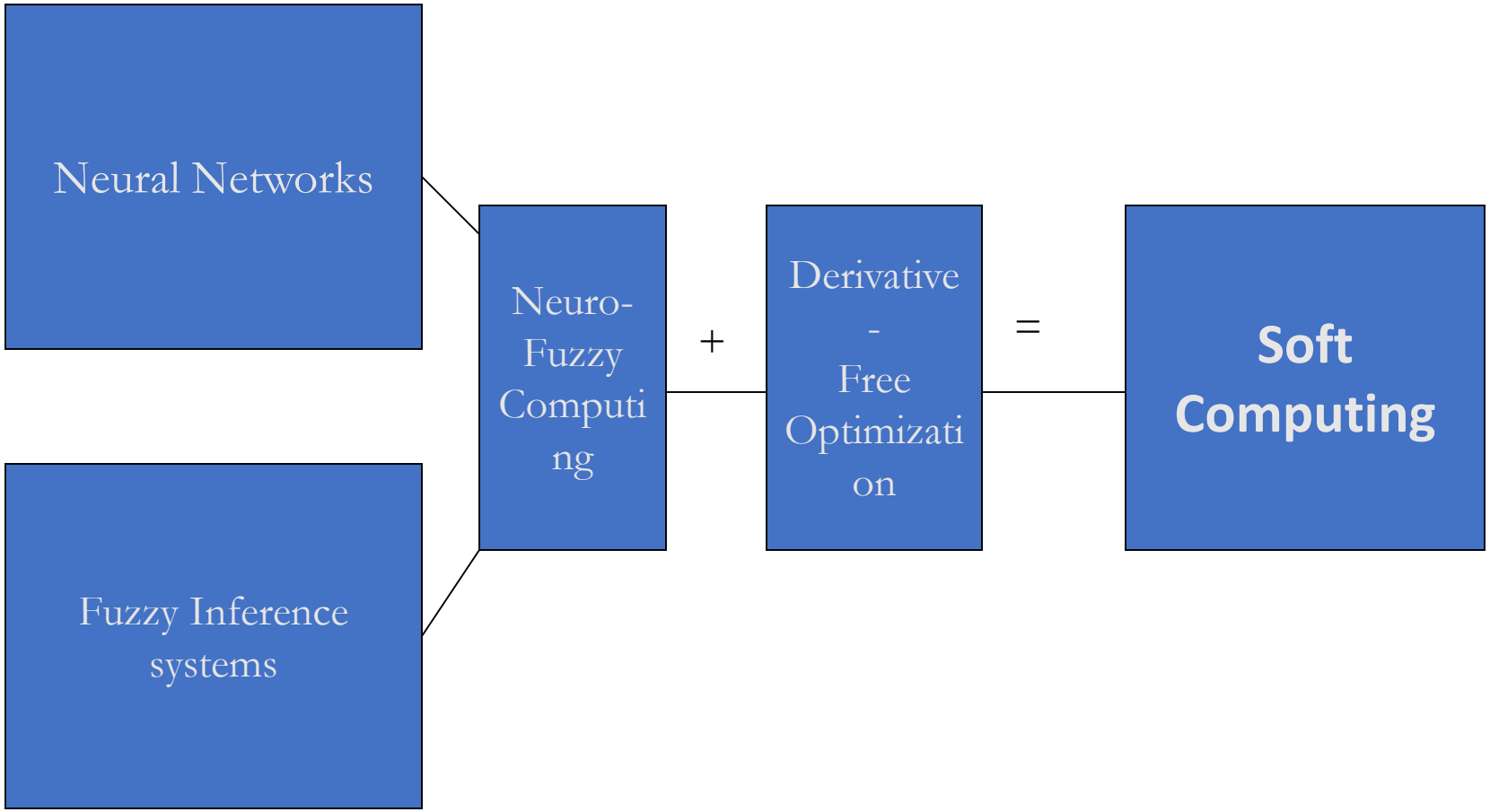
Differences between hard and soft computing

Hard Computing	Soft computing
Precisely stated analytical model	Tolerant to imprecision, uncertainty, partial truth, approximation
Based on binary logic, crisp systems, numerical analysis, crisp software	Fuzzy logic, neural nets, probabilistic reasoning.
Programs are to be written	Evolve their own programs
Two values logic	Multi valued logic
Exact input data	Ambiguous and noisy data
Strictly sequential	Parallel computations
Precise answers	Approximate answers

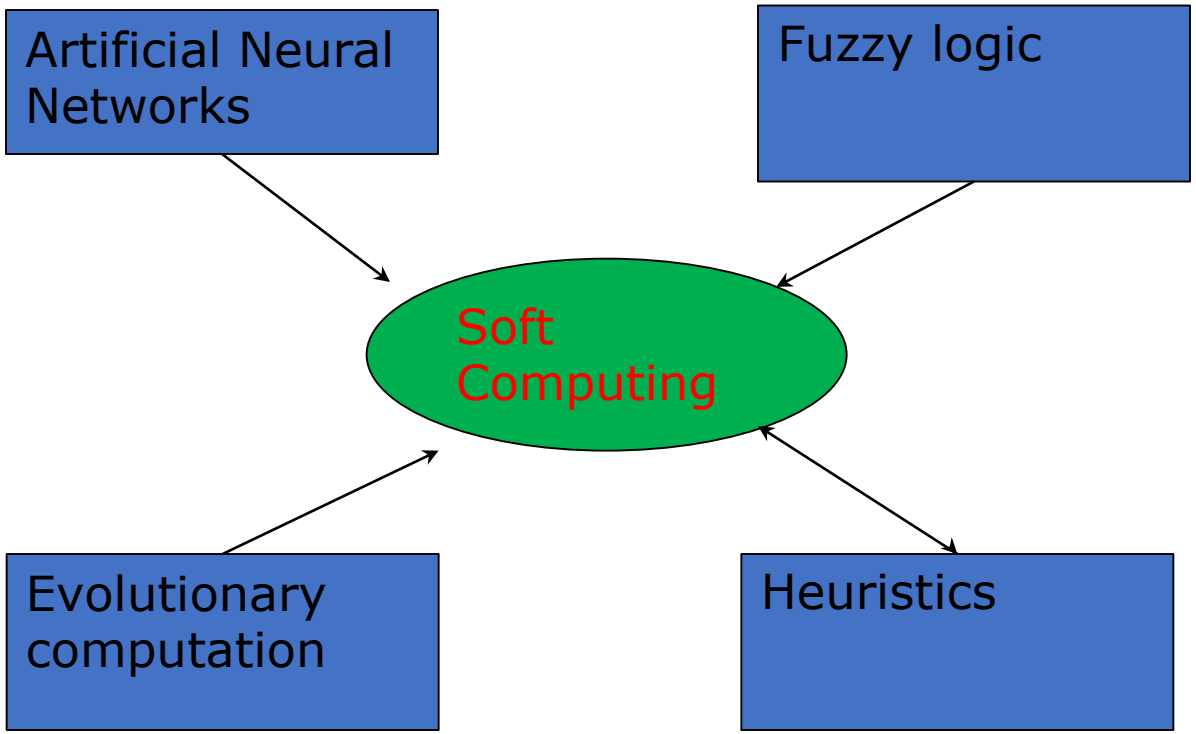
OVERVIEW OF TECHNIQUES IN SOFT COMPUTING

- Neural Networks
- Fuzzy Logic
- Genetic Algorithm
- Hybrid Systems

What is Soft computing



What is Soft computing



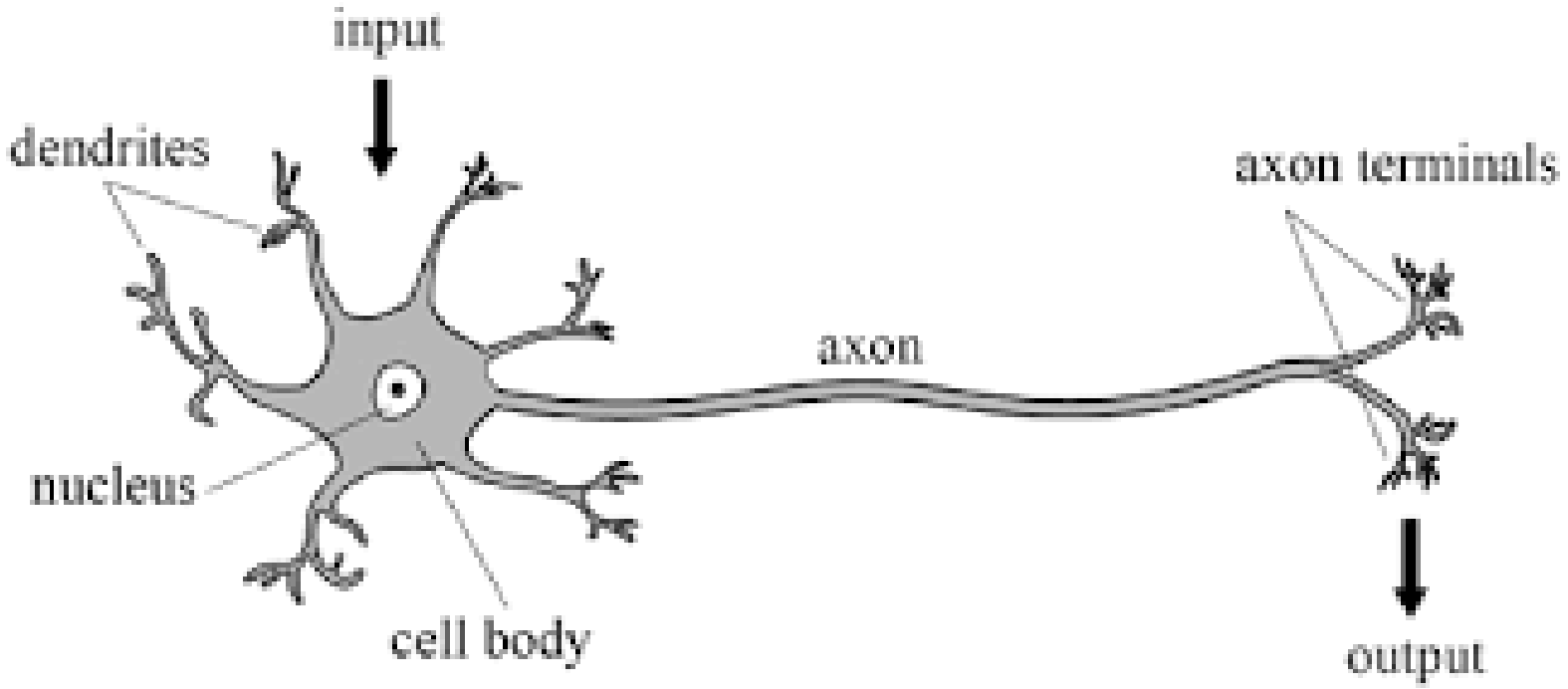
Applications

- Medical diagnosis
- Person identification / Computer vision
- Hand written character recognition
- Pattern recognition and Machine Intelligence MI
- Weather forecasting
- VLSI design
- Network optimization
- Neuro Fuzzy Systems
- Fuzzy Logic Control
- Machine Learning Applications
- Speech and Vision Recognition

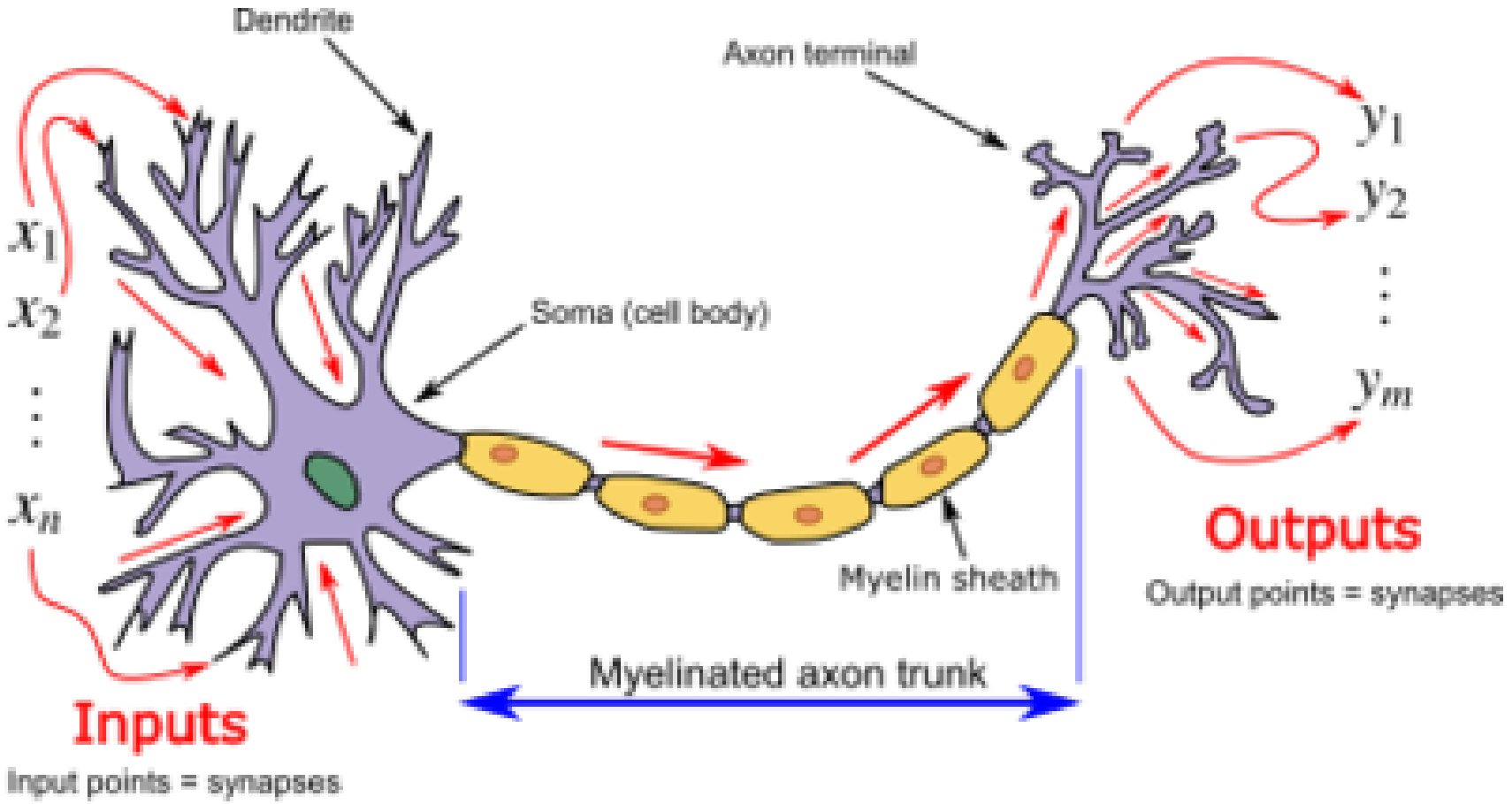
Artificial Neural Networks

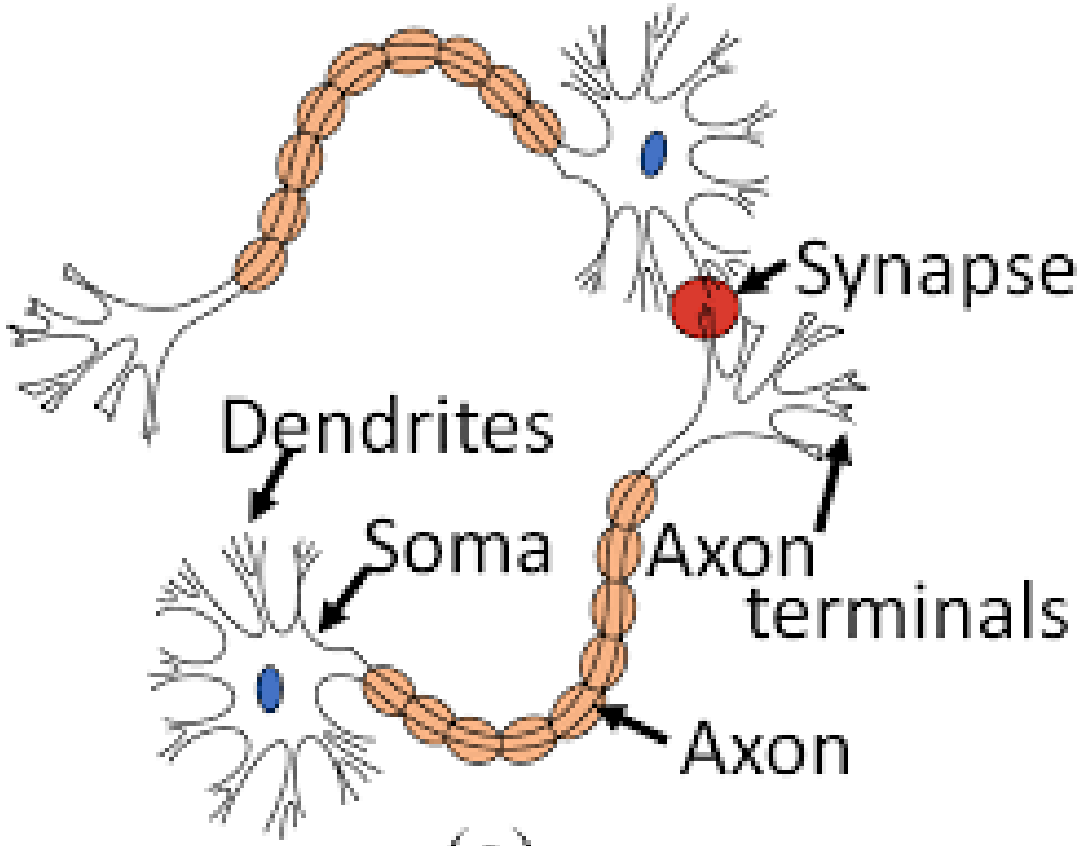
- Neural Network are constructed and implemented to model the human brain.
- Performs various tasks such as pattern-matching, classification, optimization function, approximation, vector quantization and data clustering.
- ANN possess a large number of processing elements called nodes/neurons which operate in parallel.
- Neurons are connected with others by connection link.

- dendrites: nerve fibres carrying electrical signals to the cell
- cell body: computes a non-linear function of its inputs
- axon: single long fiber that carries the electrical signal from the cell body to other neurons
- synapse: the point of contact between the axon of one cell and the dendrite of another, regulating a chemical connection whose strength affects the input to the cell.

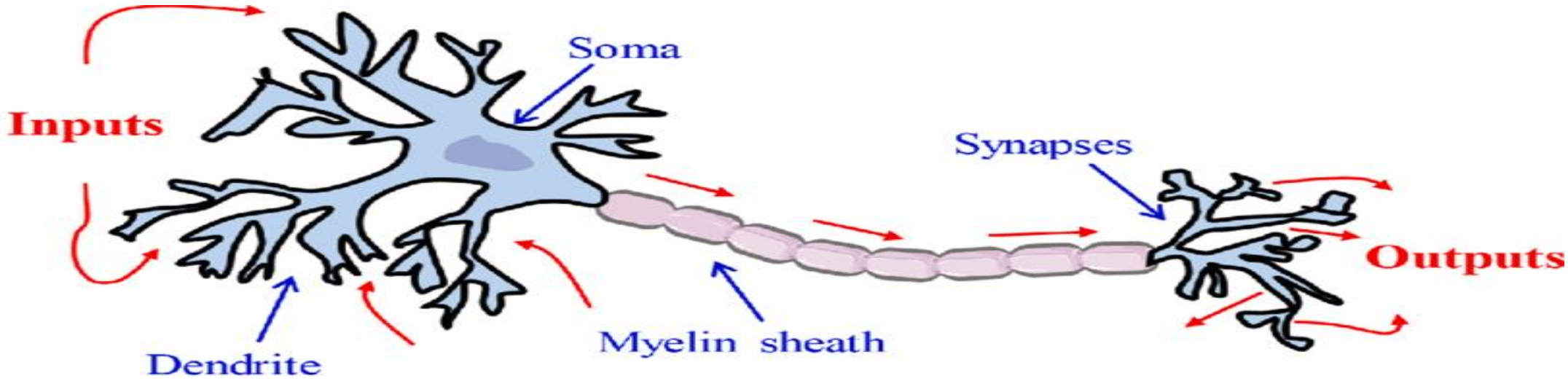


Biological Neuron

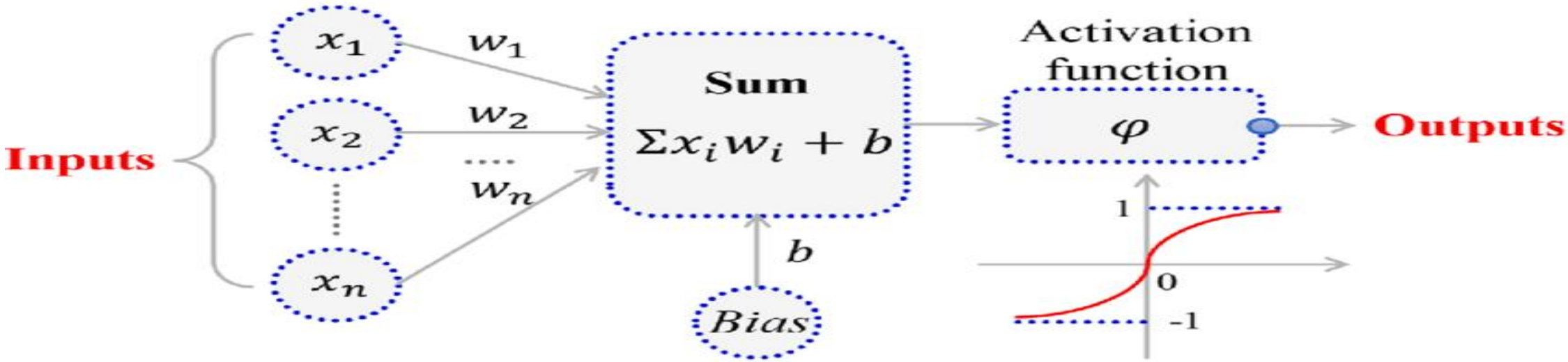




(a)

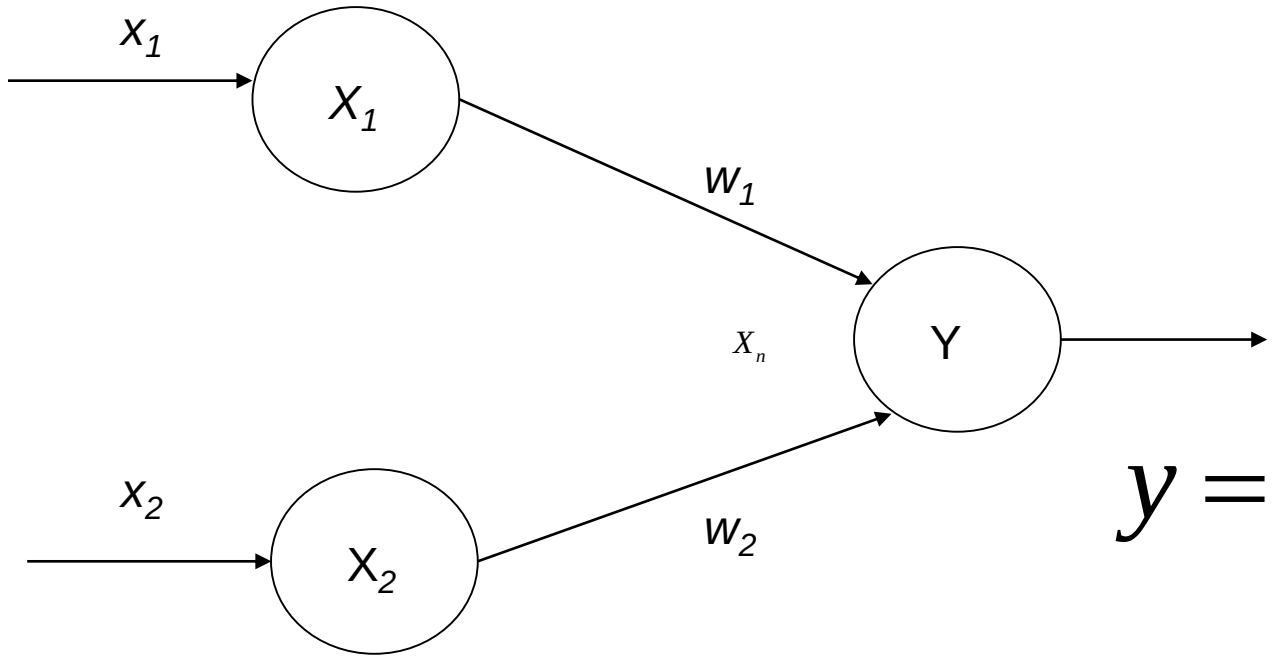


(a) Biological neuron



(b) Artificial neuron

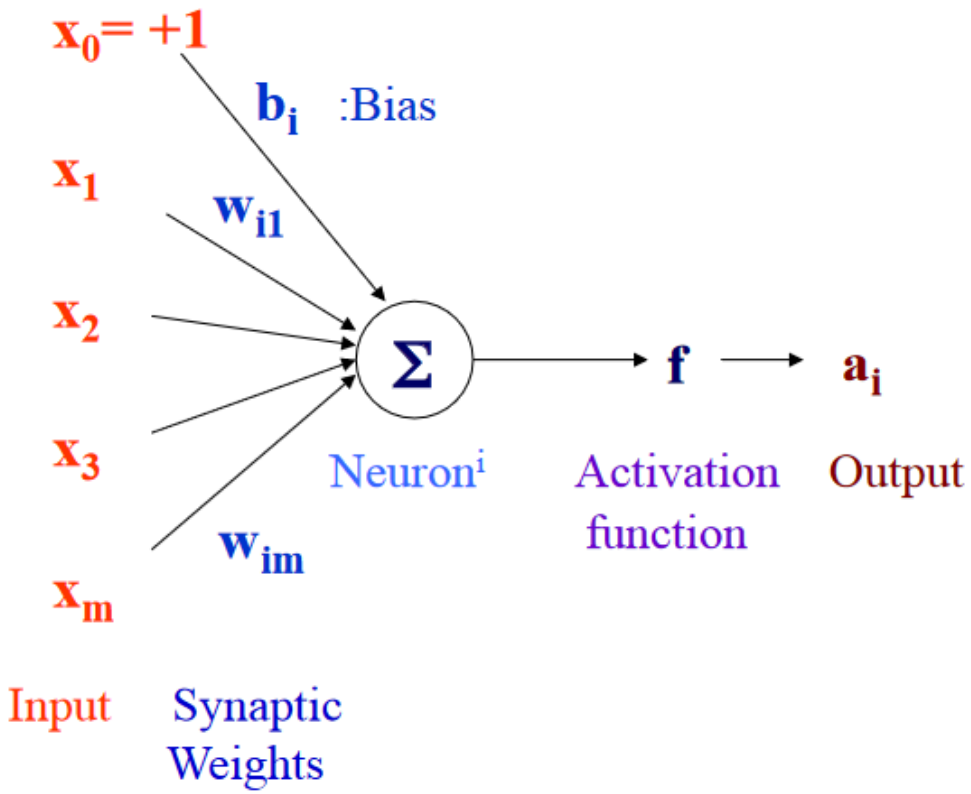
Artificial Neural Networks



$$y = f(y_{in})$$

$$y_{in} = x_1 w_1 + x_2 w_2$$

An artificial neuron:
- computes the weighted sum of its input (called its net input)
- adds its bias
- passes this value through an activation function



Terminology Relationship

Biological Neuron	Artificial Neuron
Cell	Neuron
Dendrites	Weights and Interconnections
Soma	Net Input
Axion	Output

Comparison

Term	Brain	Computer
Speed	Execution time is few milliseconds	Execution time is few nano seconds
Processing	Perform massive parallel operations simultaneously	Perform several parallel operations simultaneously. It is faster the biological neuron
Size and complexity	Number of Neuron is 10^{11} and number of interconnections is 10^{15}. So complexity of brain is higher than computer	It depends on the chosen application and network designer.
Storage capacity	i) Information is stored in interconnections or in synapse strength. ii) New information is stored without destroying old one. iii) Sometimes fails to recollect information	i) Stored in continuous memory location. ii) Overloading may destroy older locations. iii) Can be easily retrieved
Tolerance	i) Fault tolerant ii) Store and retrieve information even interconnections fails iii) Accept redundancies	i) No fault tolerance ii) Information corrupted if the network connections disconnected. iii) No redundancies
Control mechanism	Depends on active chemicals and neuron connections are strong or weak	CPU Control mechanism is very simple

BN

- Information is stored in interconnects or in synapse strength
- New information is stored without destroying old one
- Sometimes fails to recollect information

ANN

Stored in continuous memory location

Overloading may destroy older locations

Can be easily retrieved

Network Architecture

- Several different architectures for ANNs
- each with their own strengths and weaknesses.
- Some of the most common architectures include:
- **Feedforward Neural Networks**: This is the simplest type of ANN architecture, where the information flows in one direction from input to output.
- The layers are fully connected, meaning each neuron in a layer is connected to all the neurons in the next layer.
- **Recurrent Neural Networks (RNNs):** These networks have a “memory” component, where information can flow in cycles through the network.
- This allows the network to process sequences of data, such as time series or speech.

Basic models of Artificial Neural Network

- Three basic entities namely
  - The model’s synaptic interconnections
  - The learning rule for updating and adjusting the connection weights
  - Their activation function

Interconnections

- Interconnection can be defined as the way processing elements (Neuron) in ANN are connected to each other.
- Hence, the arrangements of these processing elements and geometry of interconnections are very essential in ANN.
- These arrangements always have two layers that are common to all network architectures, the Input layer and output layer where the input layer buffers the input signal, and the output layer generates the output of the network.
- The third layer is the Hidden layer, in which neurons are neither kept in the input layer nor in the output layer.
- These neurons are hidden from the people who are interfacing with the system and act as a black box to them.
- By increasing the hidden layers with neurons, the system's computational and processing power can be increased but the training phenomena of the system get more complex at the same time.

FEED BACK NETWORK

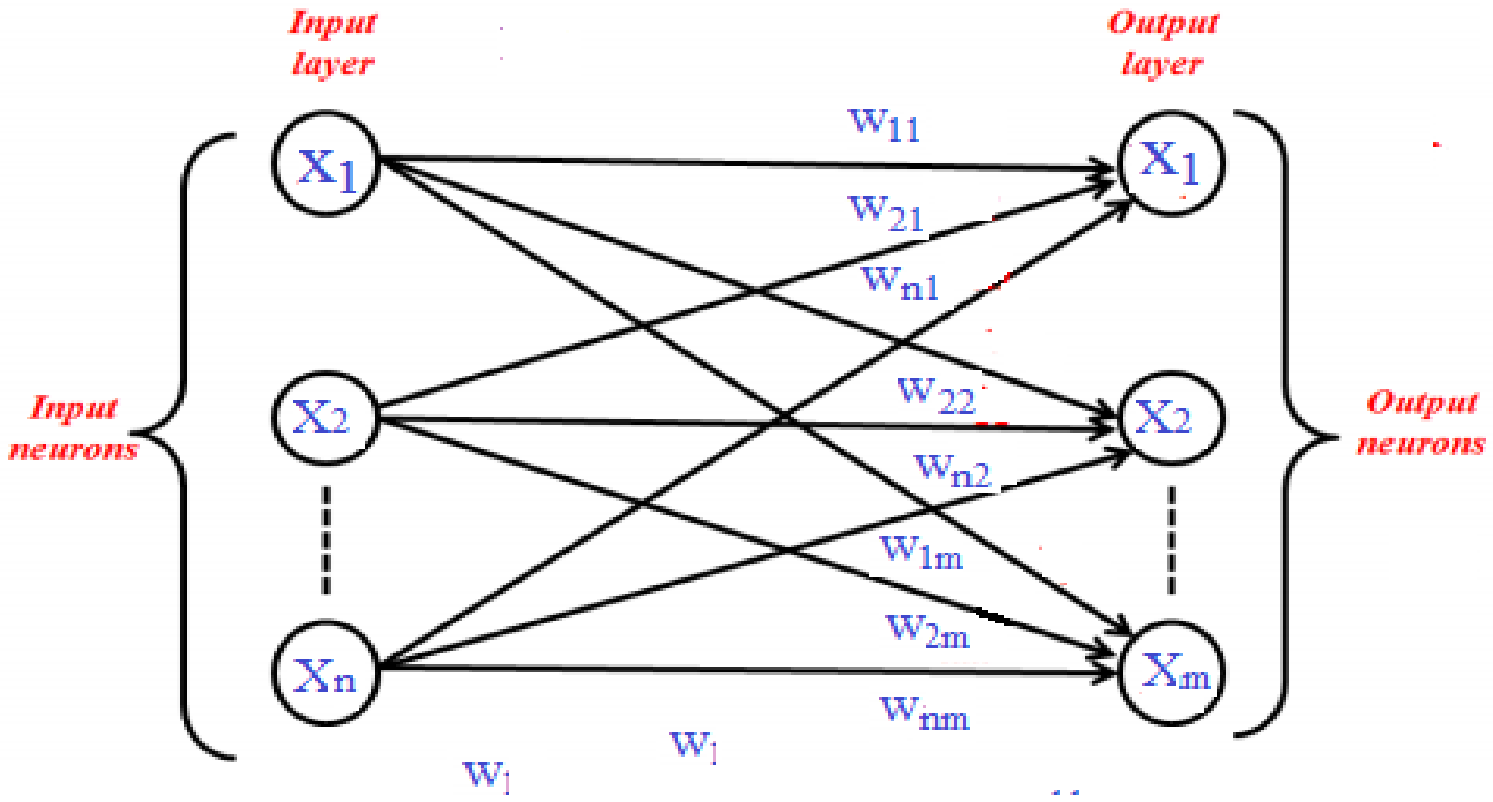
- If no neuron in the output layer is an input to a node in the same layer / proceeding layer – ***feed forward network.***
- If outputs are directed back as input to the processing elements in the same layer/proceeding layer –***feedback network.***
- If the output are directed back to the input of the same layer then it is ***lateral feedback.***
- ***Recurrent networks*** are networks with feedback networks with closed loop.

Network Architecture

- Arrangement of neurons to the layers & connection pattern formed within and between layers
  - Single layer Feed Forward network
  - Multi layer Feed Forward network
  - Single node with its own feedback
  - Single layer recurrent network
  - Multi layer recurrent network

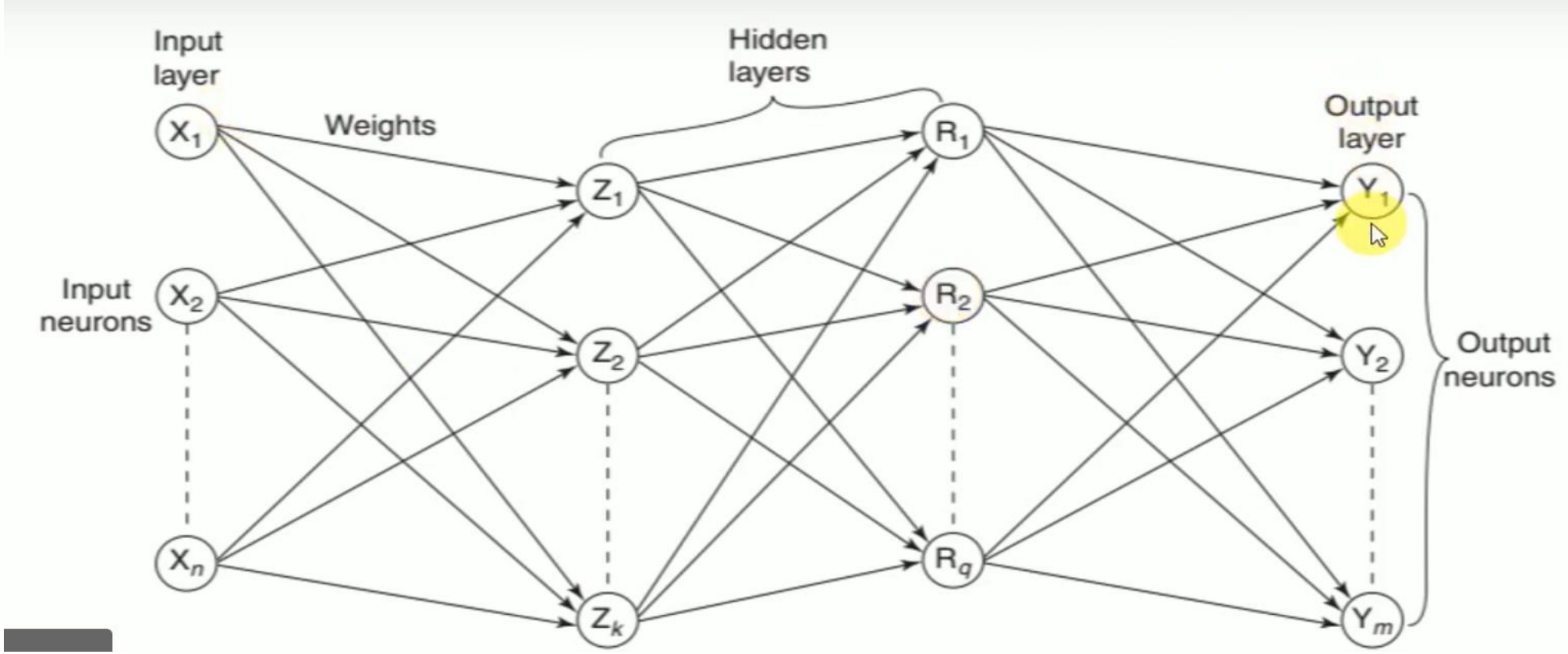
Single layer Feed Forward Network

- With only one layer, output layer, inputs are fed



- In this type of network, we have only two layers input layer and the output layer
- the input layer does not count because no computation is performed in this layer.
- The output layer is formed when different weights are applied to input nodes and the cumulative effect per node is taken.
- After this, the neurons collectively give the output layer to compute the output signals.

Multilayer Feed Forward



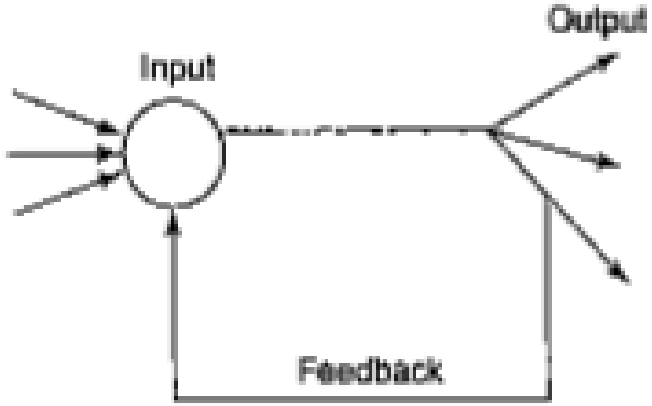
Multilayer Feed Forward

- Formed by the interconnection of several layers.
- Input layer receives input and buffers input signal.
- Output layer generated output.
- Layer between input and output is called ***hidden layer.***
- Hidden layer is internal to the network.
- Zero to several hidden layers in a network.
- More the hidden layer, more is the complexity of network, but efficient output is produced.

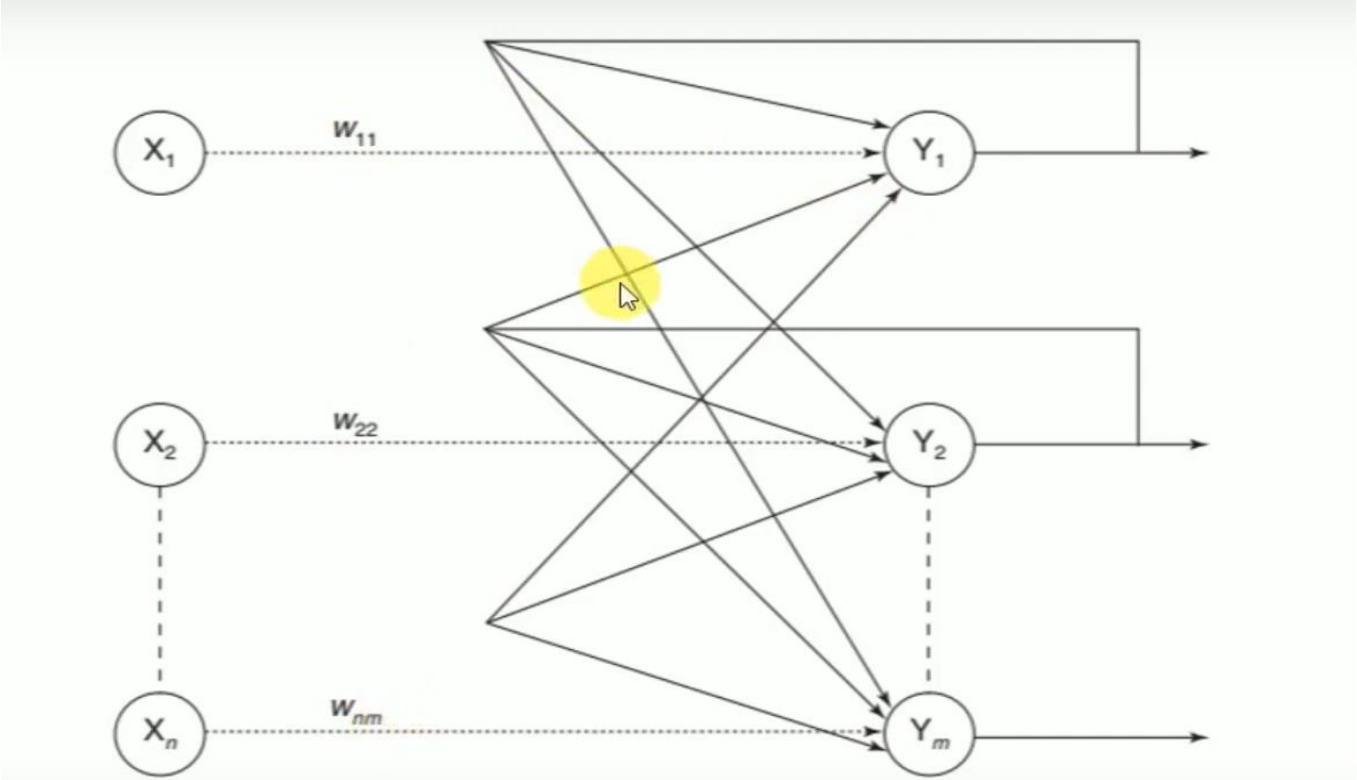
Single node with own feedback

- When output can be directed back as

Input to the same node



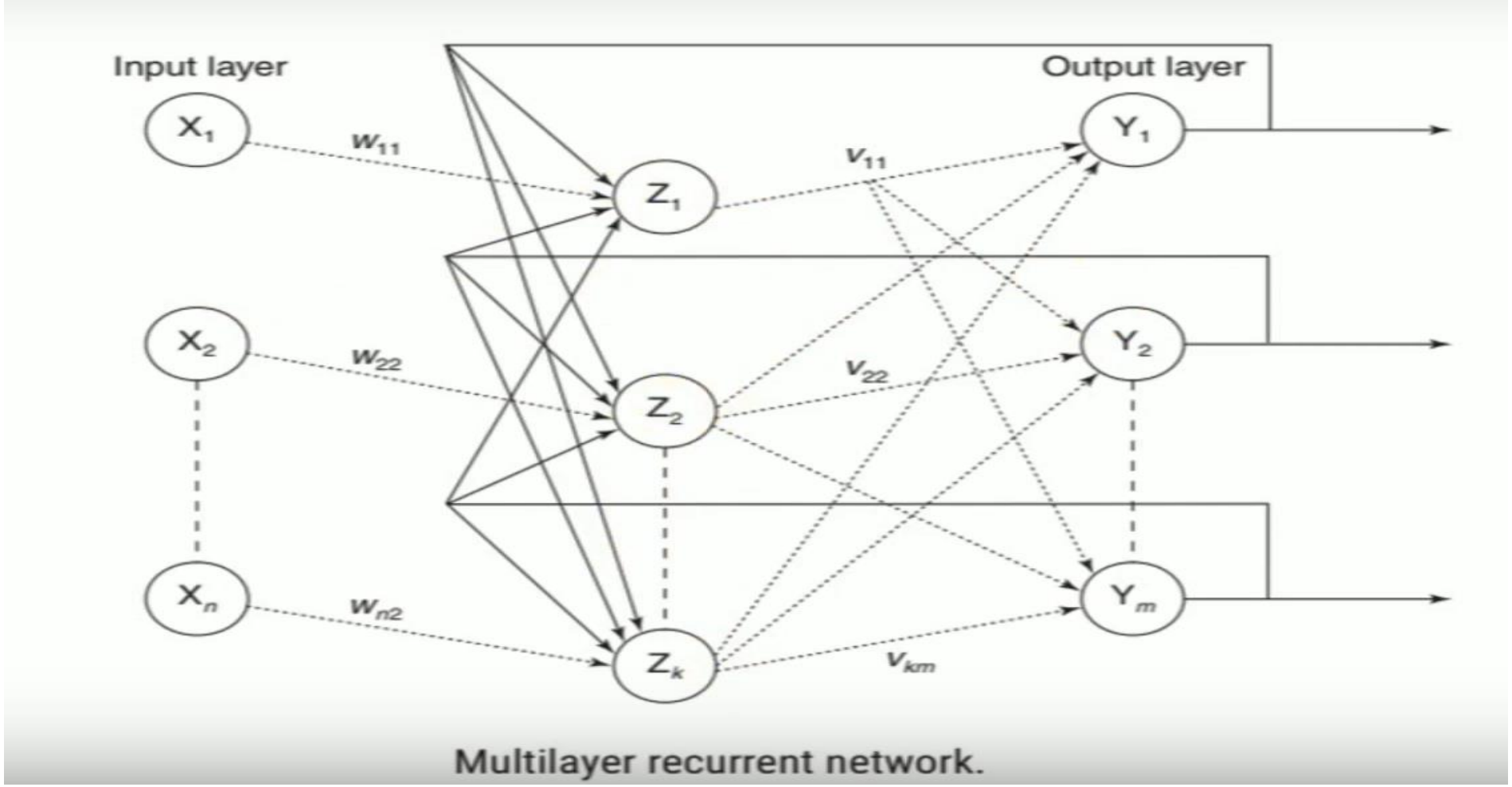
Single-layer recurrent network



Single-layer recurrent network.

- The above network is a single-layer network with a feedback connection in which the processing element’s output can be directed back to itself or to another processing element or both.
- A recurrent neural network is a class of artificial neural networks where connections between nodes form a directed graph along a sequence.
- Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.

Multilayer recurrent network

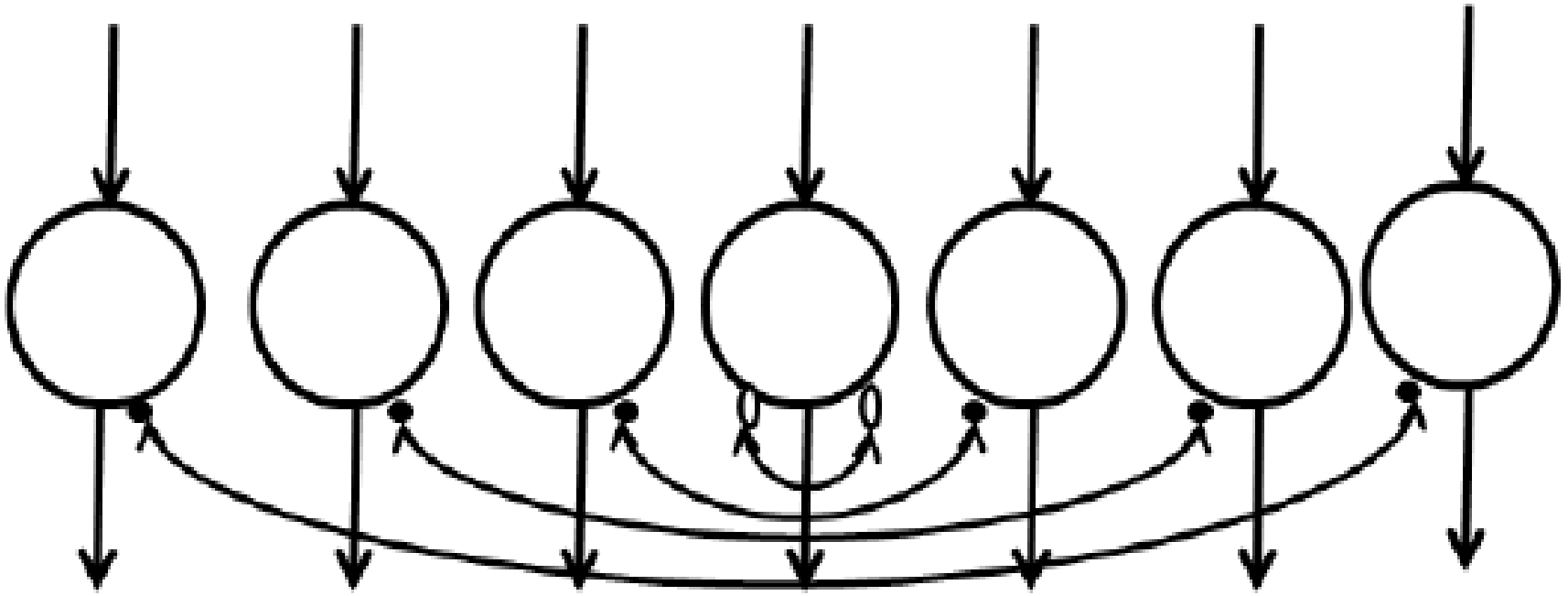


Multilayer recurrent network.

- In this type of network, processing element output can be directed to the processing element in the same layer and in the preceding layer forming a multilayer recurrent network.
- They perform the same task for every element of a sequence, with the output being dependent on the previous computations.
- Inputs are not needed at each time step.
- The main feature of a Recurrent Neural Network is its hidden state, which captures some information about a sequence.

Lateral Inhibition Structure

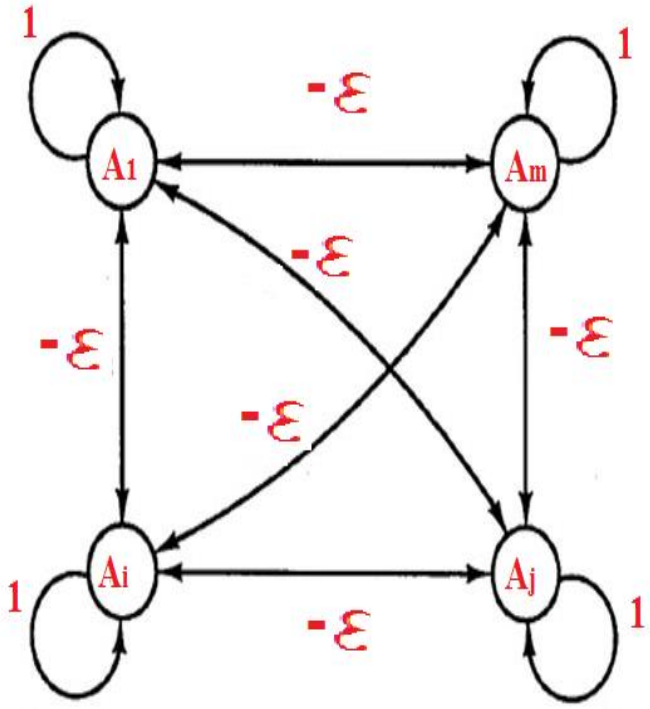
- Signals ending at shaded circles are inhibitory and ending at unfilled circles are excitatory



- lateral inhibition is a structure of a network in which neurons inhibit their neighbors
- Neurons in the network receive input $F_i(x)$, where x is a coordinate of a particular neuron.
- They send inhibitory connections to neighboring neurons with weight distributed according to a function $W(y - x)$.

COMPETITIVE NETS

- Nodes compete against each other by sending out inhibiting signals to each other
- Done by setting the weights between different nodes to be negative
- During training process also the weights remains fixed in these competitive networks. The idea of competition is used among neurons for enhancement of contrast in their activation functions.
- Maxnet -fully connected network with each node connecting to every other nodes, including itself



Learning

- Process to adapt to stimulus by making proper parameter adjustments, resulting in production of desired response
- Structure learning-focuses on change in network structure
- Parameter learning –updates connecting weights

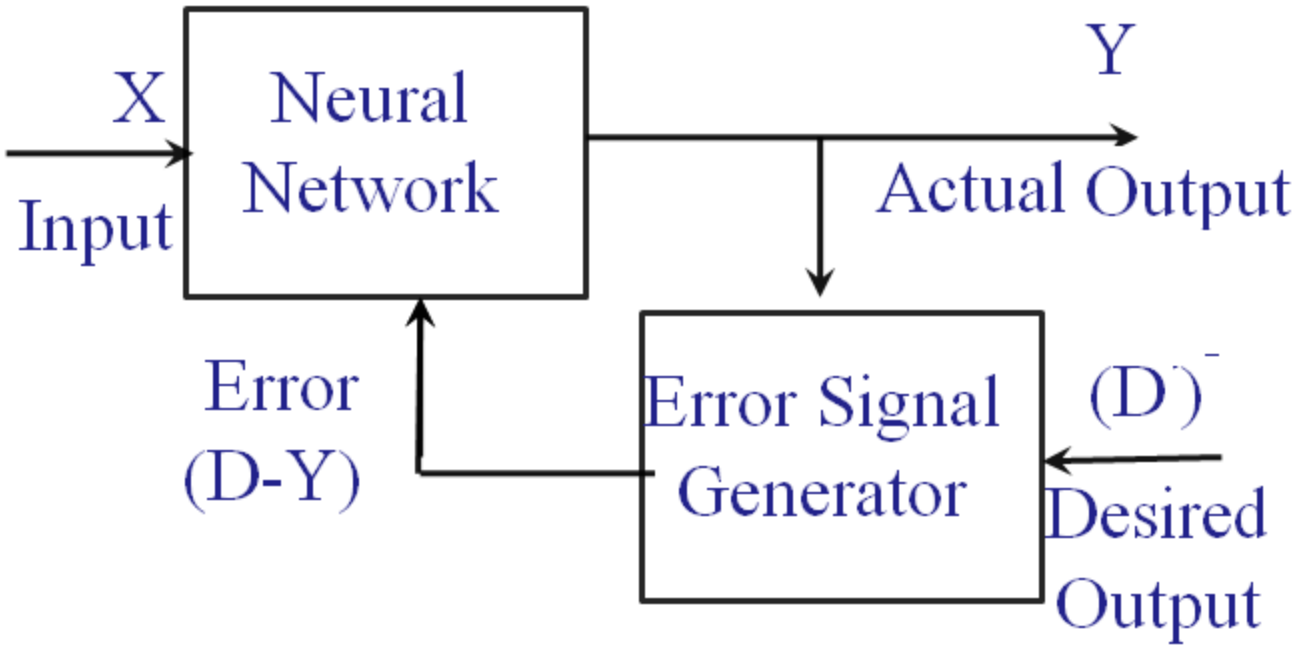
Types of Learning

- Supervised- category of machine learning that uses labeled datasets to train algorithms to predict outcomes and recognize patterns.
- Unsupervised- type of machine learning that learns from data without human supervision.
- Reinforcement

Supervised Learning

- Supervisor know the answer-label
  - Precise information about expected output
- Training is required
  - Input vector and target-training pair
- Mistakes are corrected
  - difference of output and target as error signal
- Error signal for weight adjustment

Working


X
Input
Neural Network
Y
Actual Output
Error
(D-Y)
Error Signal Generator
(D)
Desired Output

Sorting Fruits

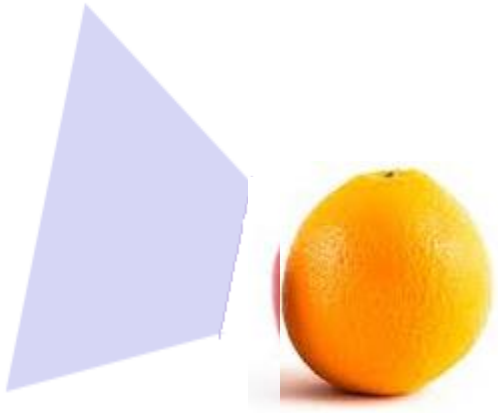
Orange

Apple

Colour
Shape
Size

Colour
Shape
Size

Sorting Fruits



- ✔ Learning with the help of a teacher.
- ✔ Example : learning process of a small child.
  - ✔ Child doesn’t know read/write.
  - ✔ Their each & every action is supervised by a teacher
- ✔ In ANN, each input vector requires a corresponding target vector, which represents the desired output.
- ✔ The input vector along with target vector is called ***training pair.***
- ✔ The input vector results in output vector.
- ✔ The actual output vector is compared with desired output vector.
- ✔ If there is a difference means an error signal is generated by the network.
- ✔ It is used for adjustment of weights until actual output matches desired output.

This is a shirt we used to wear.

Color: green
Size : Large
Type : Formal

Is this a shirt ?

Ok

Color: Green, red
Size : Large, small
Type : Formal, casual

This is also a shirt

Is this a shirt ?

Color: Green, red, yellow
Size : Large, small, medium
Type : Formal, casual

Ok

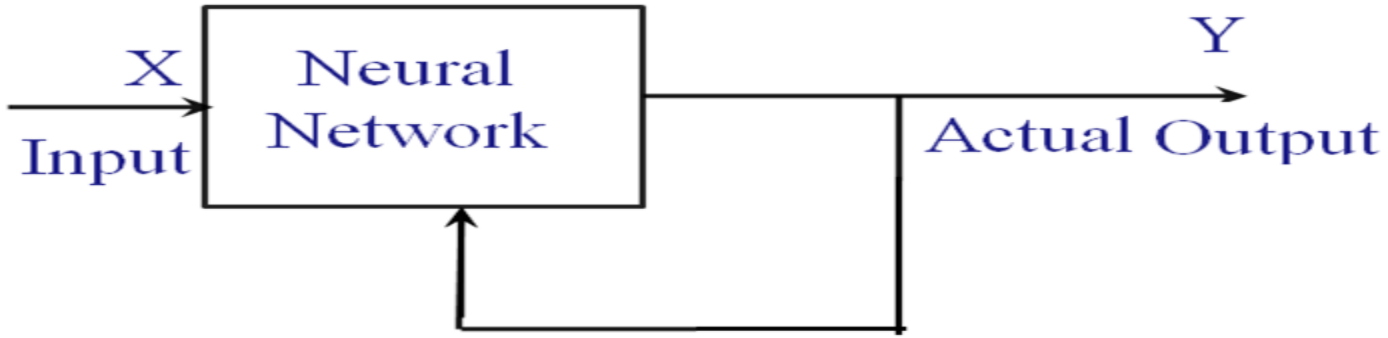
This is also a shirt

Yes, these are all shirts

Now I can identify every shirt

Unsupervised

- type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision
- Learns by itself



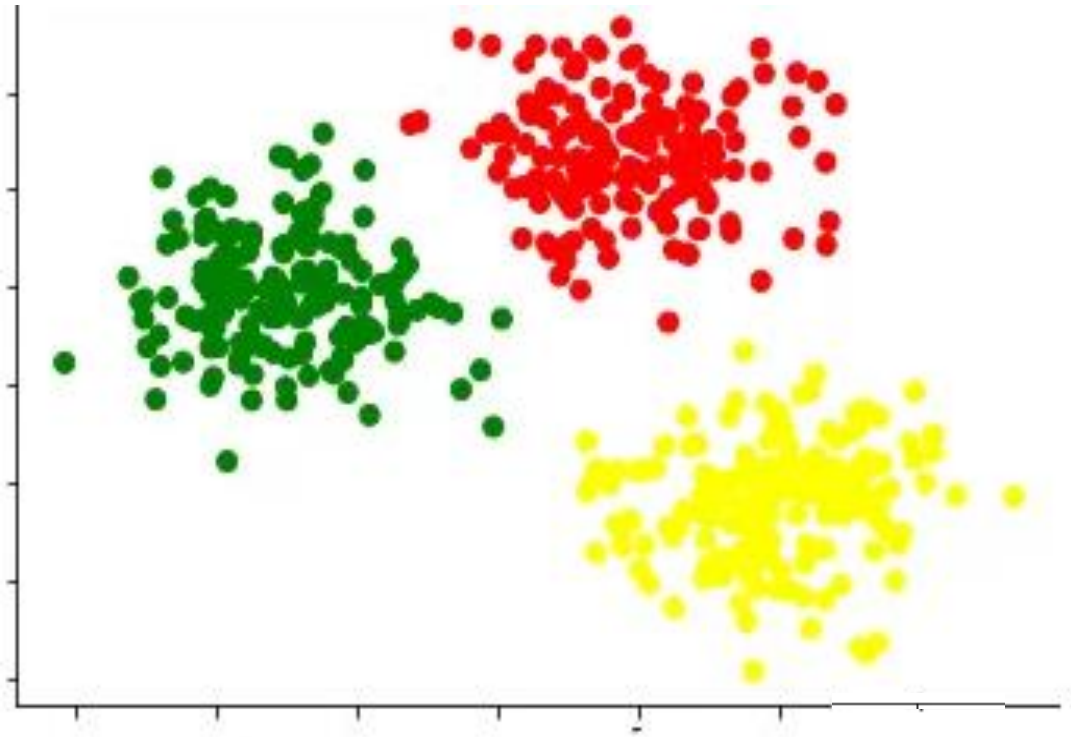
UNSUPERVISED LEARNING

- ✔ Learning is performed without the help of a teacher.
- ✔ Example: tadpole – learn to swim by itself.
- ✔ In ANN, during training process, network receives input patterns and organize it to form clusters.
- ✔ From the Fig. it is observed that no feedback is applied from environment to inform what output should be or whether they are correct.
- ✔ The network itself discover patterns, regularities, features/ categories from the input data and relations for the input data over the output.
- ✔ Exact clusters are formed by discovering similarities & dissimilarities so called as ***self – organizing.***

Working

- Explore data to understand patterns and associations
- Clustering- Similar elements are grouped
- Training-input patterns are organized to clusters
- For new input- class is determined, if not found,new class generated
- Network to discover features and categories
- Self organising- determining exact clusters

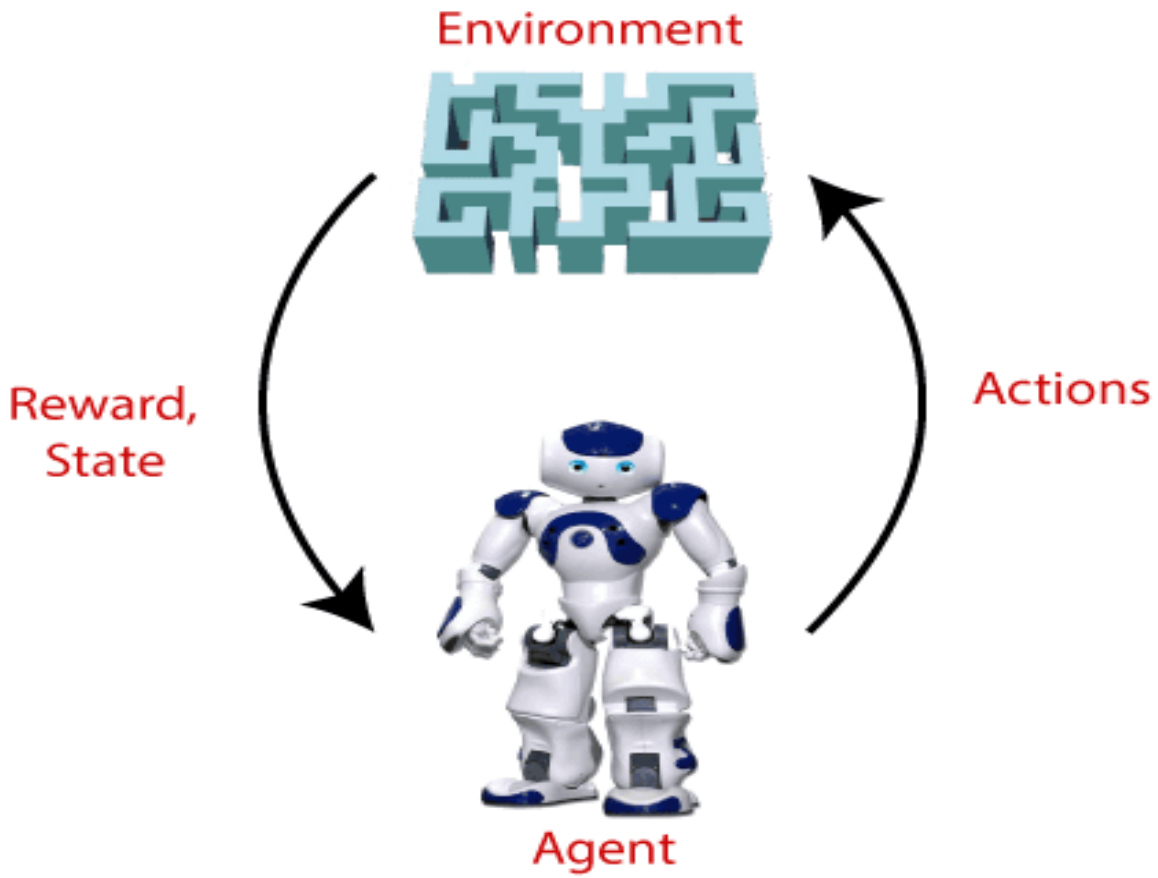
Clustering

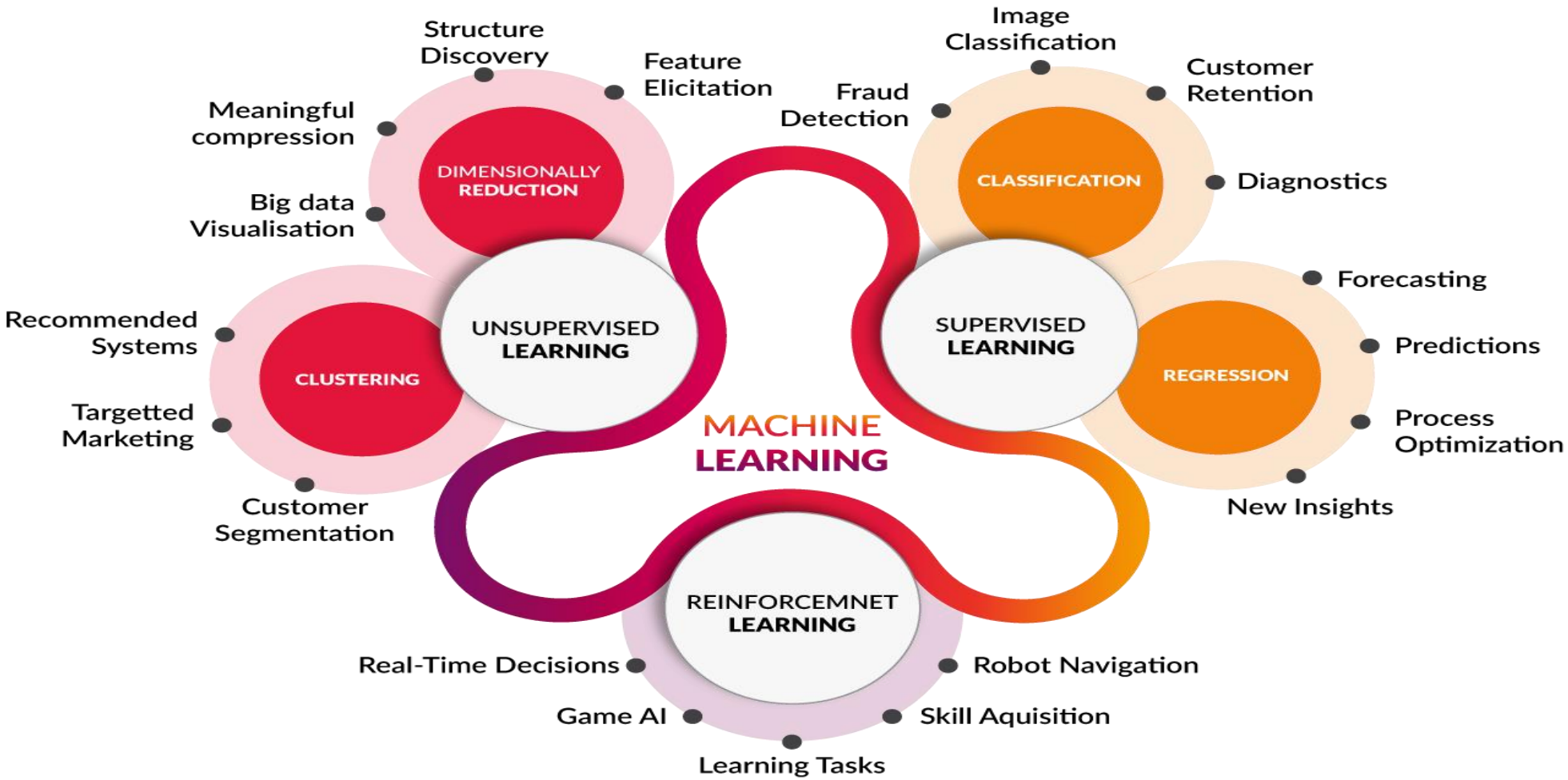


Reinforcement Learning

- Reinforcement Learning is a feedback-based Machine learning technique
- Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- An agent learns to behave in an environment by performing the actions and seeing the results of actions.
- For each good action, the agent gets positive feedback,
- for each bad action, the agent gets negative feedback or penalty.

Reinforcement Learning




MACHINE LEARNING
UNSUPERVISED LEARNING
DIMENSIONALLY REDUCTION
Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
CLUSTERING
Recommended Systems
Targetted Marketing
Customer Segmentation
SUPERVISED LEARNING
CLASSIFICATION
Image Classification
Customer Retention
Fraud Detection
Diagnostics
REGRESSION
Forecasting
Predictions
Process Optimization
New Insights
REINFORCEMNET LEARNING
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

Difference

Supervised Learning	Unsupervised LEarning	Reinforcement Learning
works with the labelled data and here the output data patterns are known to the system	deals with unlabeled data where the output is based on the collection of perceptions.	the agent interacts with the environment in discrete steps.
highly supervised. goal is to generate formula based on input and output values	not supervised, find an association between input values and group them	depends on the agent in determining the output. an agent learn through delayed feedback by interacting with the environment.
input data in Supervised Learning in labelled data	the data is unlabeled	data is not predefined in Reinforcement Learning.
predicts based on a class type	discovers underlying patterns.	the learning agent works as a reward and action system
maps labelled data to known output	explore patterns and predict the output	follows a trial and error method.

ACTIVATION FUNCTIONS

- ✔ The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it.
- ✔ This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.
- ✔ An integration function (*f*) is associated with input of processing element.
- ✔ The nonlinear activation function is used to ensure that a neuron’s response is bounded

✔ Several activation functions are there.

1. Identity function:

it is a linear function which is defined as

$$f(x) = x \text{ for all } x$$

The output is same as the input.

2. Binary step function

it is defined as

$$f(x) = \begin{cases} 1 & if\ x \geq \theta \\ 0 & if\ x < \theta \end{cases}$$

where θ represents threshold value.

It is used in single layer nets to convert the net input to an output that is binary. (0 or 1)

3. Bipolar step function:

It is defined as

$$f(x) = \begin{cases} 1 & if\ x \geq \theta \\ -1 & if\ x < \theta \end{cases}$$

where θ represents threshold value

used in single layer nets to convert the net input to an output that is bipolar (+1 or -1).

4. Sigmoid function

used in Back propagation nets.

Two types:

a) binary sigmoid function

-logistic sigmoid function or unipolar sigmoid function.

-it is defined as

The range of sigmoid function is 0 to 1.

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

b) Bipolar sigmoid function

where λ- steepness parameter and the sigmoid range is between -1 and +1.

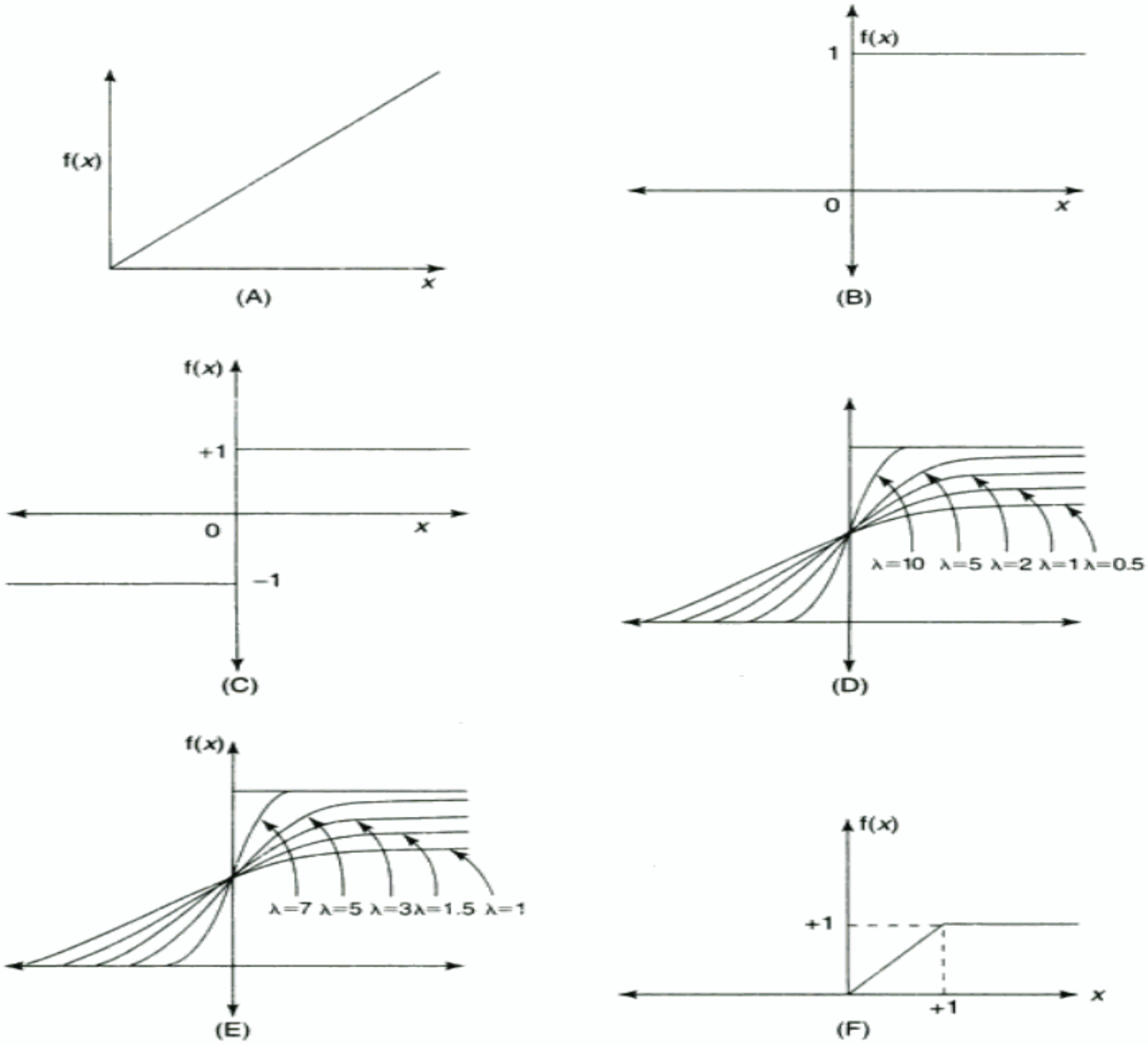
-

$$f(x) = \frac{2}{1 + e^{-\lambda x}} - 1 = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}}$$

5. Ramp function

$$f(x) = \begin{cases} 1 & if\ x > 1 \\ x & if\ 0 \leq x \leq 1 \\ 0 & if\ x < 0 \end{cases}$$

The graphical representation of all these function is given in the upcoming Figure



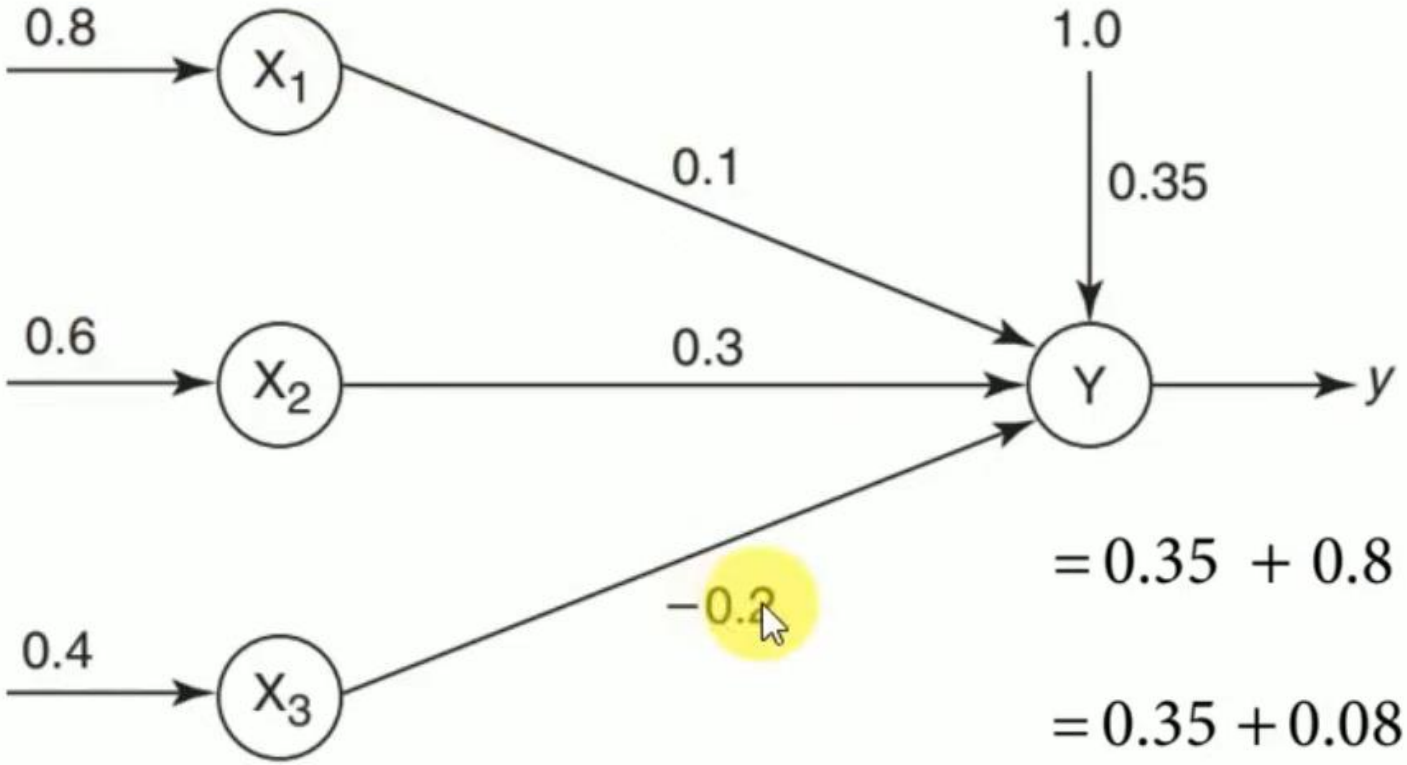
2-15 Depiction of activation functions: (A) identity function; (B) binary step function; (C) bipolar step function; (D) binary sigmoidal function; (E) bipolar sigmoidal function; (F) ramp function.

Function	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Bipolar step		$f(x) = \begin{cases} -1 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Binary Sigmoid		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Bipolar Sigmoid		$f(x) = \frac{2}{1+e^{-\lambda x}} - 1$	$f'(x) = \frac{\lambda}{2}\left[1 + f(x)\right]\left[1 - f(x)\right]$
Tanh		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
RELU		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Ramp		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } 0 \leq x < 1 \\ 1 & \text{for } x > 1 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } 0 \leq x < 1 \\ 0 & \text{for } x > 1 \end{cases}$

- Calculate the output y of a three input neuron with bias. The input feature vector (x1,x2,x3) is (0.8, 0.6, 0.4) and weight values are [w1,w2,w3,b]= [0.1, 0.3, -0.2, 0.35], Use

(i) binary sigmoidal (ii) bipolar sigmoidal as the activation function

Sigmoid activation function



$$= 0.35 + 0.8 \times 0.1 + 0.6 \times 0.3 + 0.4 \times (-0.2)$$

$$= 0.35 + 0.08 + 0.18 - 0.08 = 0.53$$

$$y_{in} = b + \sum_{i=1} x_i w_i$$

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3$$

- Binary Sigmoid Activation Function

$$y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} = \frac{1}{1+e^{-0.53}} = 0.625$$

- Bipolar Sigmoid Activation Function

$$y = f(y_{in}) = \frac{2}{1+e^{-y_{in}}} - 1 = \frac{2}{1+e^{-0.53}} - 1 = 0.259$$

IMPORTANT TERMINOLOGIES

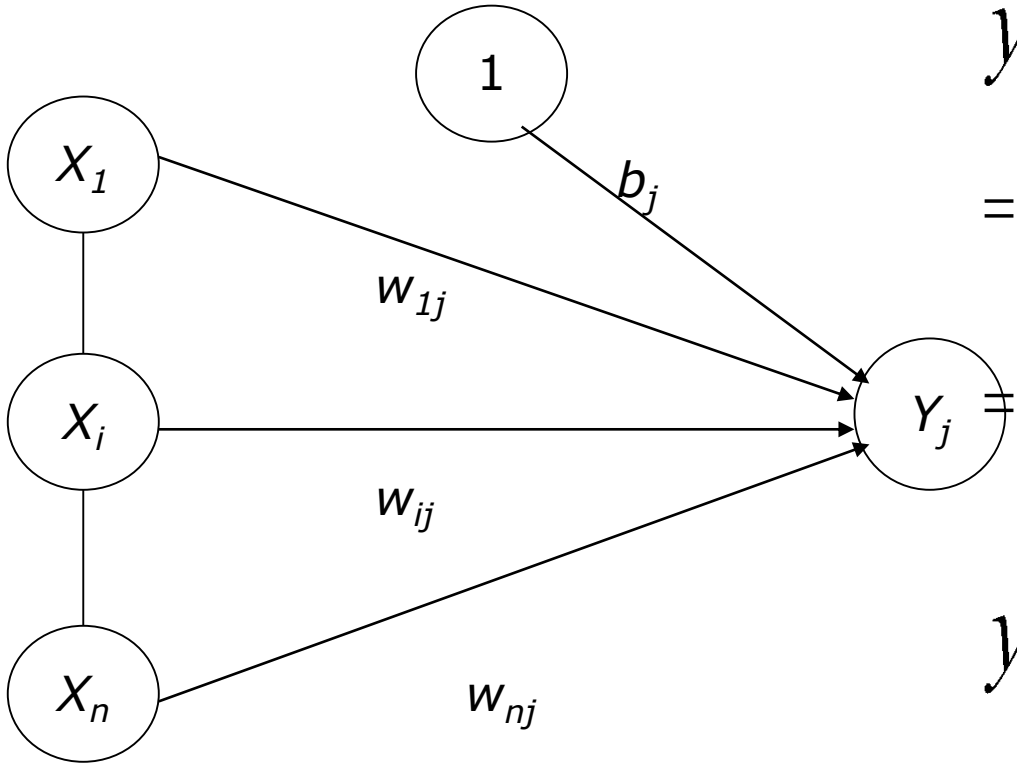
- **Weight**
  - The weight contain information about the input signal.
  - It is used by the net to solve the problem.
  - It is represented in terms of matrix & called as *connection matrix*.
  - If weight matrix W contains all the elements of an ANN, then the set of all W matrices will determine the set of all possible information processing configuration.
  - The ANN can be realized by finding an appropriate matrix W.
  - Weight encode long-term memory (LTM) and the activation states of network encode short-term memory (STM) in a neural network.

Weight matrix

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ w_3^T \\ . \\ . \\ . \\ . \\ . \\ w_n^T \end{bmatrix} = \begin{bmatrix} w_{11} w_{12} w_{13} \cdots w_{1m} \\ w_{21} w_{22} w_{23} \cdots w_{2m} \\ \cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots \\ w_{n1} w_{n2} w_{n3} \cdots w_{nm} \end{bmatrix}$$

Weights contd…

- w_{ij} _is the weight from processing element *"i"* (source node) to processing element *"j"* (destination node)



$$y_{inj} = \sum_{i=0}^{n} x_i w_{ij}$$

$$= x_0 w_{0j} + x_1 w_{1j} + x_2 w_{2j} + \ldots + x_n w_{nj}$$

$$= w_{0j} + \sum_{i=1}^{n} x_i w_{ij}$$

$$y_{inj} = b_j + \sum_{i=1}^{n} x_i w_{ij}$$

Bias

- Bias has an impact in calculating net input.
- Bias is included by adding x_0 to the input vector x.
- The net output is calculated by

$$y_{in_j} = \sum_{i=0}^{n} x_i w_{ij}$$

$$y_{in_j} = b_j + \sum_{i=0}^{n} x_i w_{ij}$$

- The bias is of two types
  - Positive bias
    - » Increase the net input
  - Negative bias
    - » Decrease the net input

Threshold

- It is a set value based upon which the final output is calculated.
- Calculated net input and threshold is compared to get the network output.
- The activation function of threshold is defined as

$$f(net) = \begin{cases} 1 & if\ net \geq \theta \\ -1 & if\ net < \theta \end{cases}$$

- where θ is the fixed threshold value

- **Learning rate**
  - Denoted by α.
  - Control the amount of weight adjustment at each step of training.
  - The learning rate range from 0 to 1.
  - Determine the rate of learning at each step
- **Momentum Factor**
  - Convergence is made faster if a momentum factor is added to the weight updation process.
  - model converges when additional training will not improve the model
  - Done in back propagation network.
- **Vigilance parameter**
  - Denoted by ρ.
  - Used in Adaptive Resonance Theory (ART) network.
  - Used to control the degree of similarity.
  - Ranges from 0.7 to 1 to perform useful work in controlling the number of clusters.

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment

learning rate

weight gradient

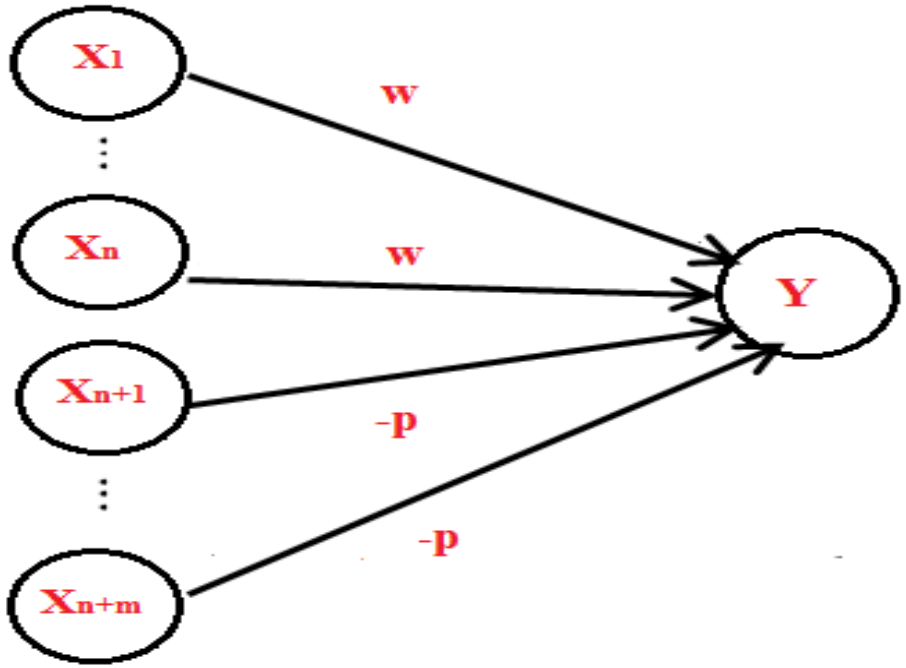
$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor

weight increment, previous iteration

McCulloch –Pitts Neuron

- McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified computational model in 1943- M-P Neuron
- Excitatory and inhibitory
- All same w, p
- Threshold important
- Binary activation



McCulloch –Pitts Neuron

- Firing of the output of the neuron is based upon the threshold on the activation function

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } y_{in} < \theta \end{cases}$$

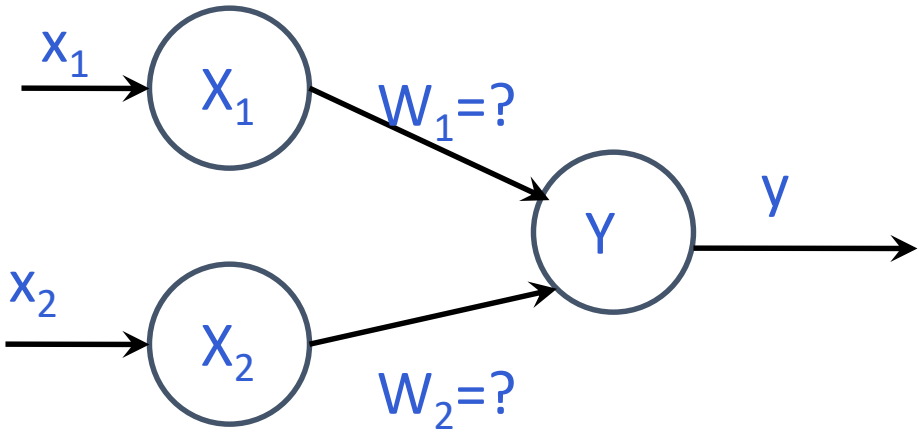
- Inhibition absolute- nw-p $\leq \theta$

  n –no: of neurons in the i/p layer

  w – positive weight

  p – negative weight

Implementation of AND Gate



x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

If we assume w1=1, w2=1,

For inputs (0,0) yin= x1w1+x2w2=0*1+0*1=0

(0,1) yin= x1w1+x2w2=0*1+1*1=1

(1,0) yin= x1w1+x2w2=1*1+0*1=1

(1,1) yin= x1w1+x2w2=1*1+1*1=2

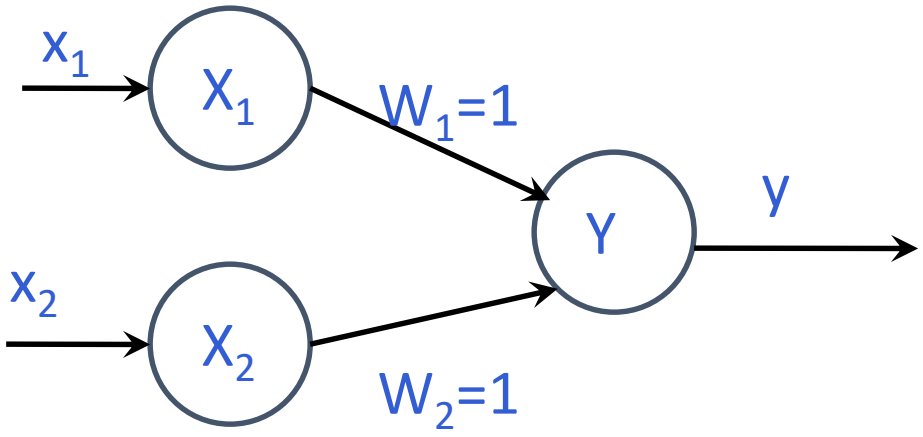
nw-p $\leq \theta$ $\theta = 2*1+0 = 2$

- This can also be obtained by

$$\theta \geq nw - p$$

- Here, n = 2, w = 1 (excitatory weights) and p = 0 (no inhibitory weights).
- Substituting these values in the above-mentioned equation we get $\theta \geq 2 \times 1 - 0 \Rightarrow \theta \geq 2$

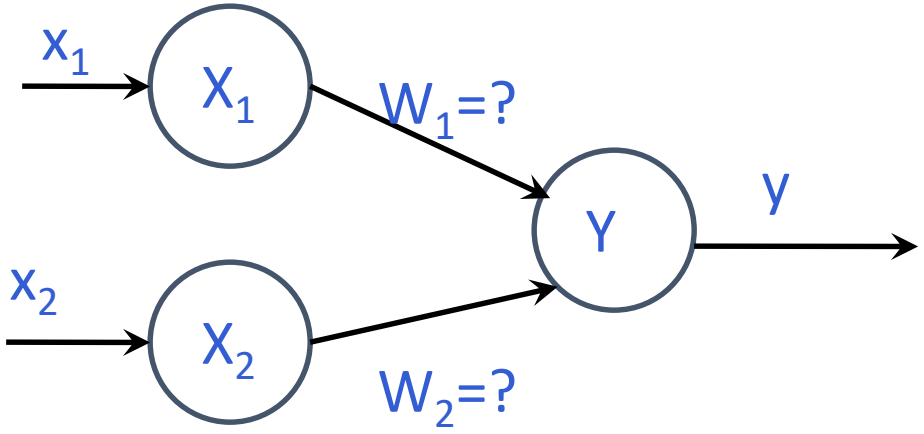
Implementation of AND Gate



x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	2

$$y = f(y_{in}) = \begin{cases} 1 & \textit{if } y_{in} \geq 2 \\ 0 & \textit{if } y_{in} < 2 \end{cases}$$

Implementation of AndNot Gate



x1	x2	y
0	0	0
0	1	0
1	0	1
1	1	0

If we assume w1=1, w2=-1,

For inputs (0,0) yin= x1w1+x2w2=0*1+0*-1=0

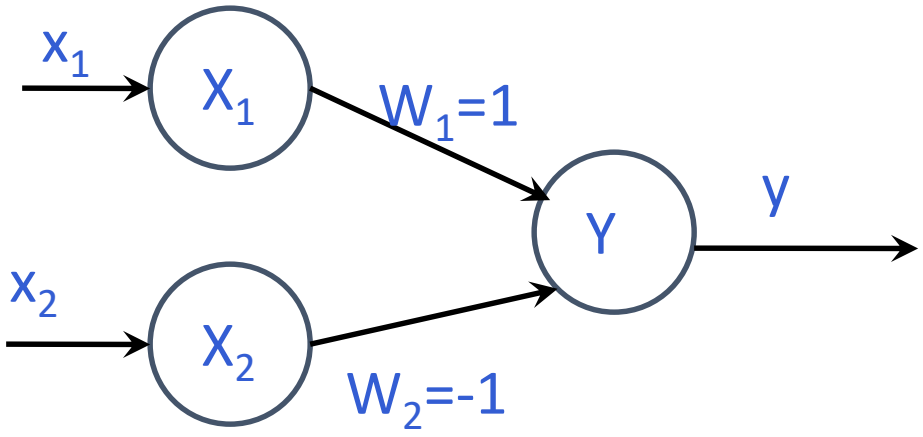
(0,1) yin= x1w1+x2w2=0*1+1*-1=-1

(1,0) yin= x1w1+x2w2=1*1+0*-1=1

(1,1) yin= x1w1+x2w2=1*1+1*-1=0

nw-p $\leq \theta$ nw-p $= 2*1-1 = 1 \leq \theta$ $\theta = 1$

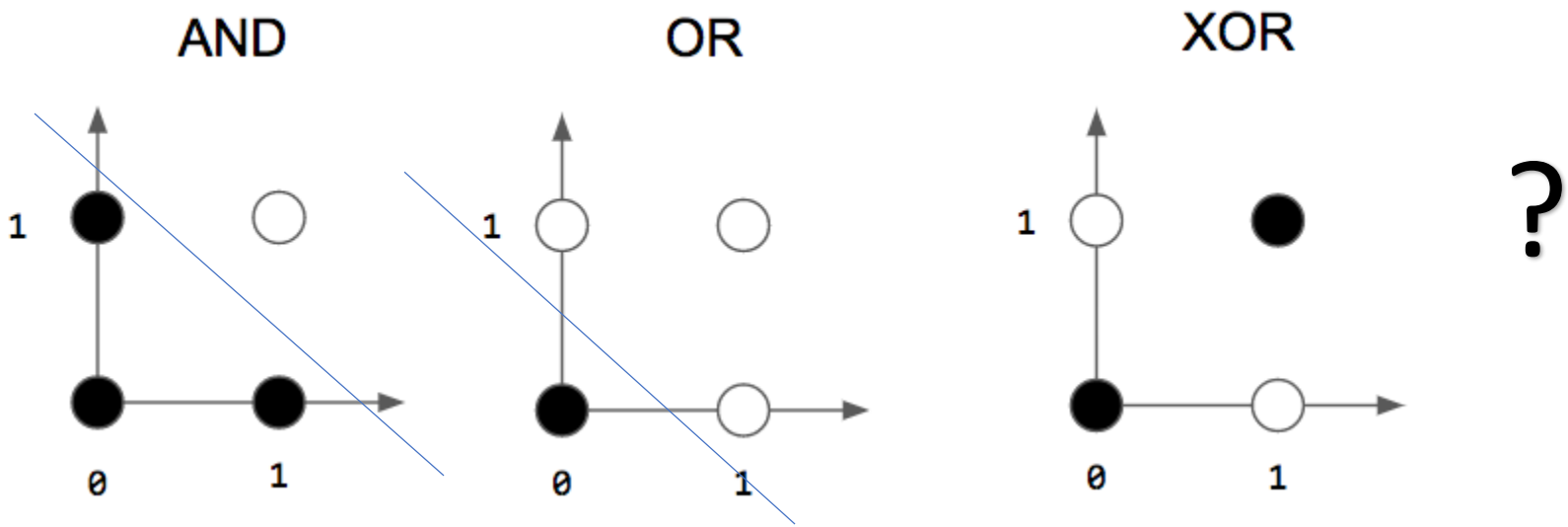
Implementation of AndNot Gate(A AND NOT B)



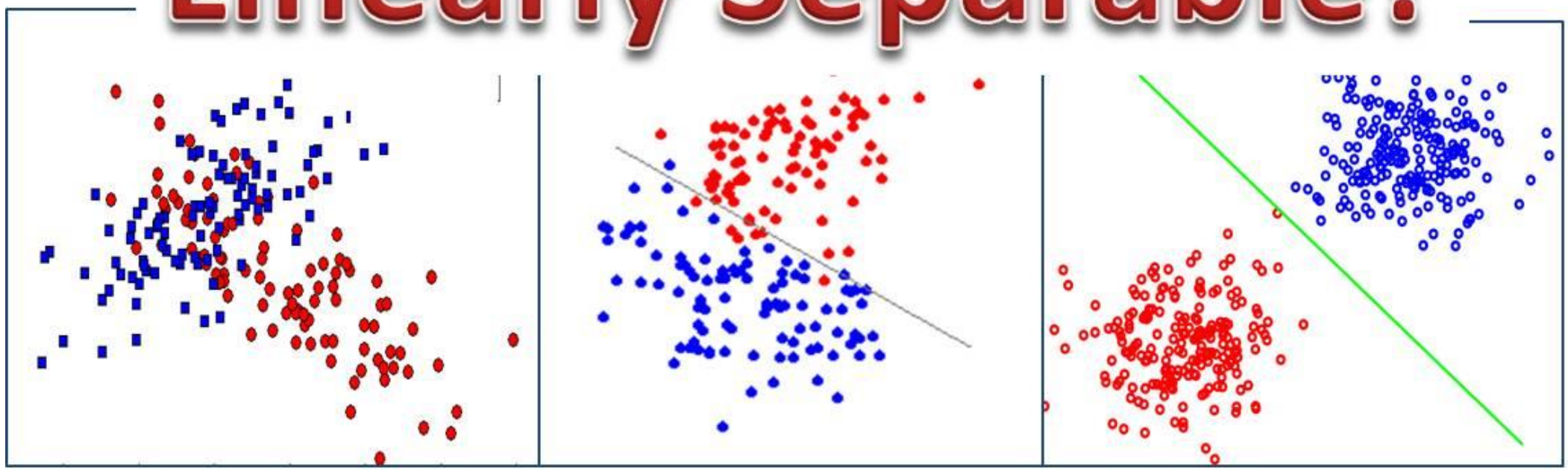
x1	x2	y
0	0	0
0	1	-1
1	0	1
1	1	0

$$y = f(y_{in}) = \begin{cases} 1 & \textit{if } y_{in} \geq 1 \\ 0 & \textit{if } y_{in} < 1 \end{cases}$$

More Gates

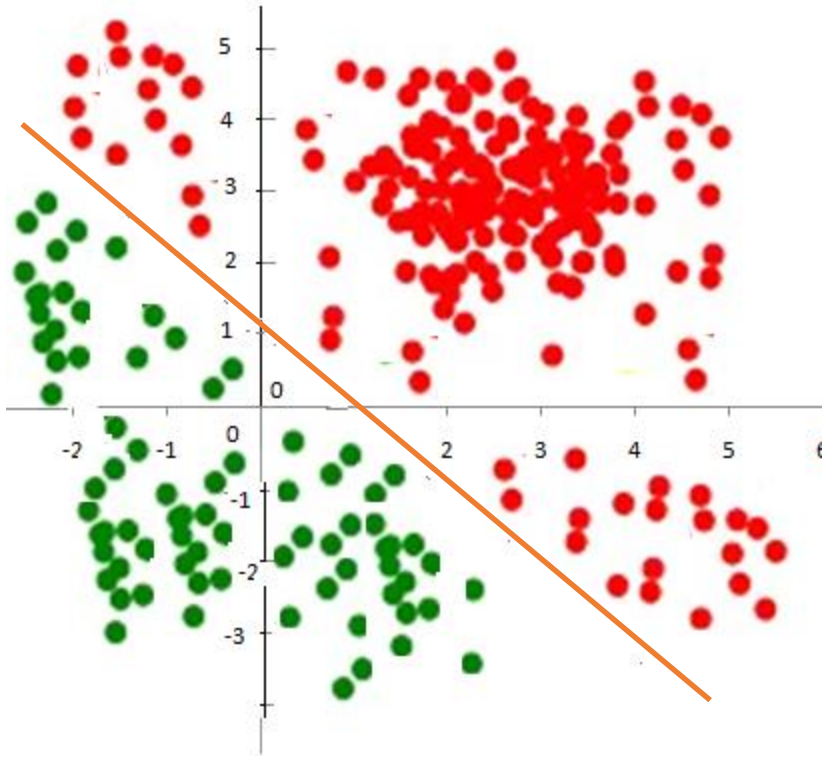


Which one is Linearly Separable?



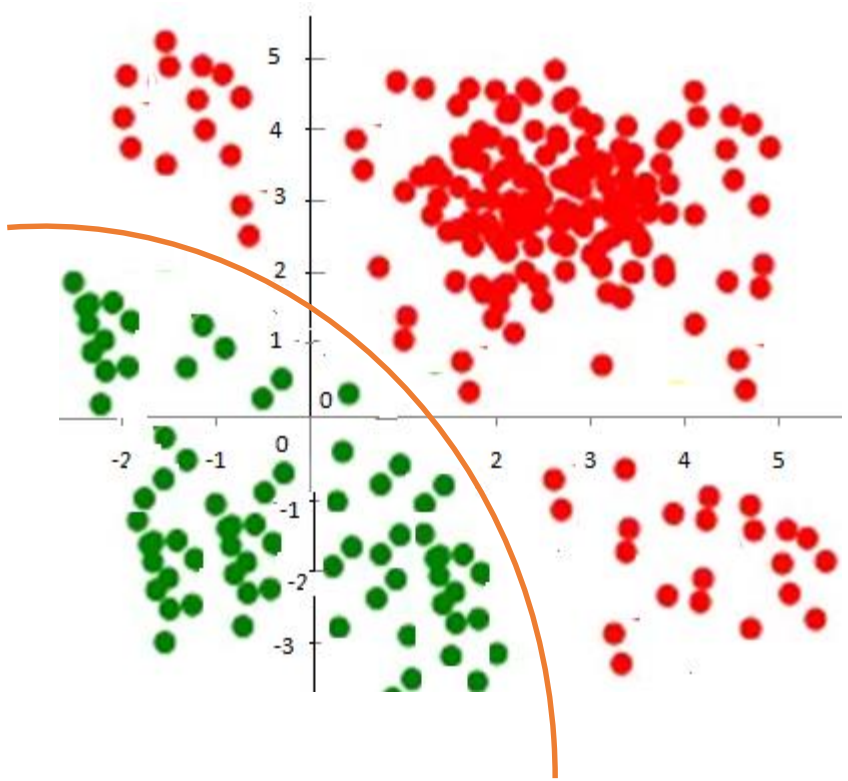
Set 1 Set 2 Set 2

Separability

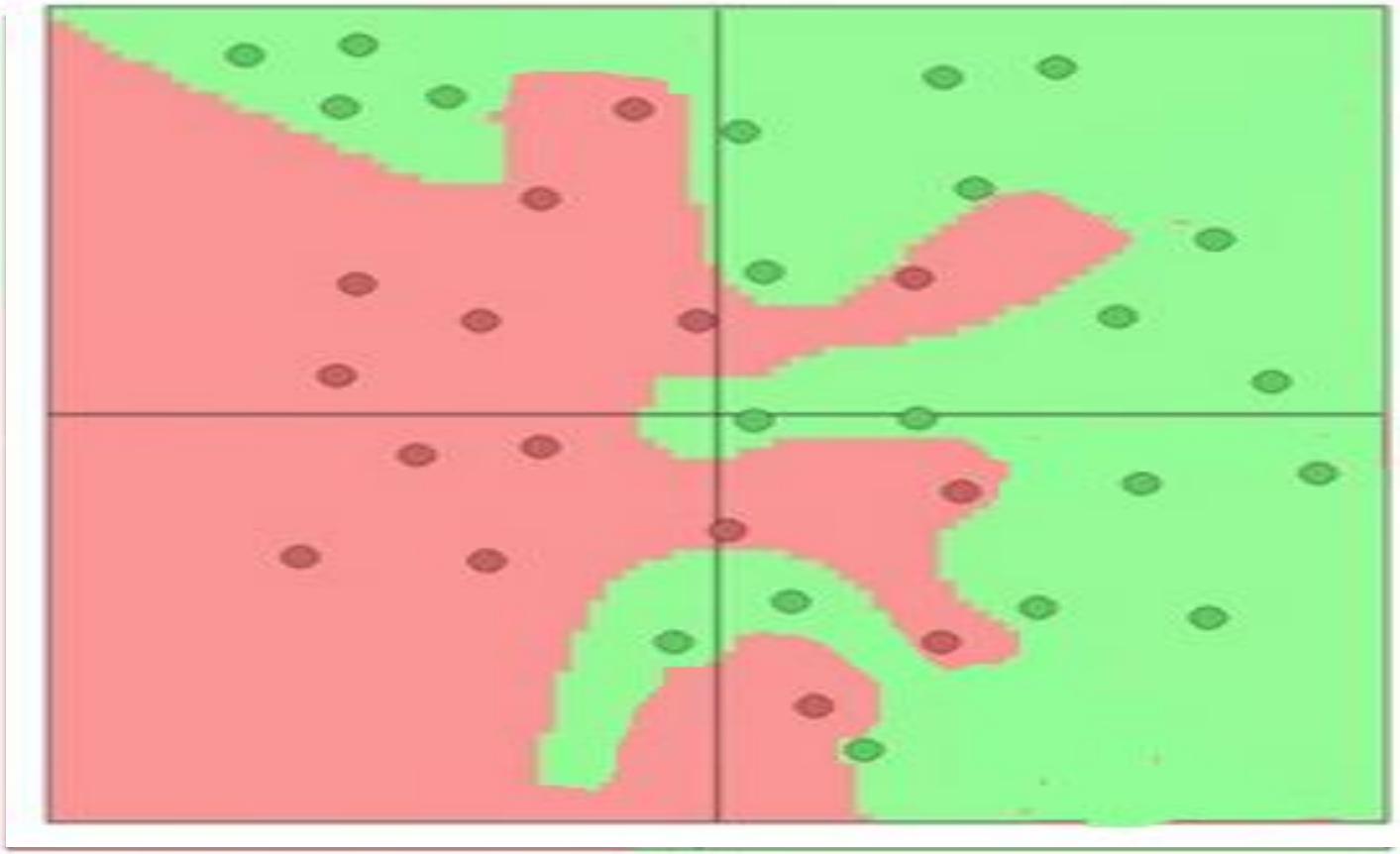


- Linear separability

Separability

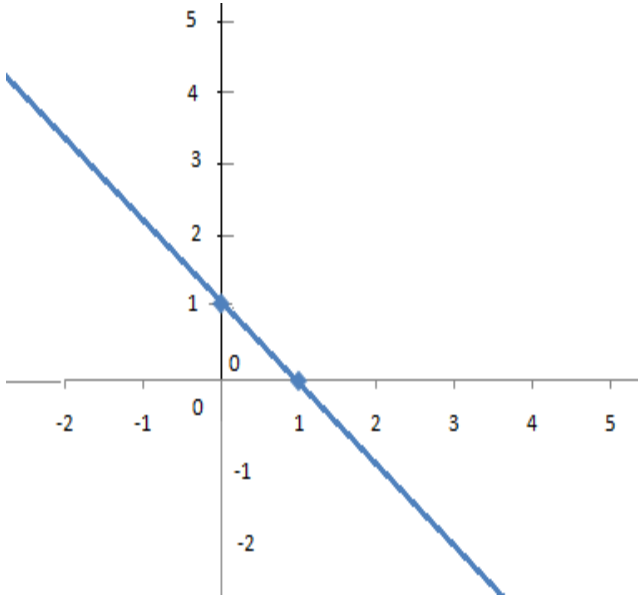


Separability

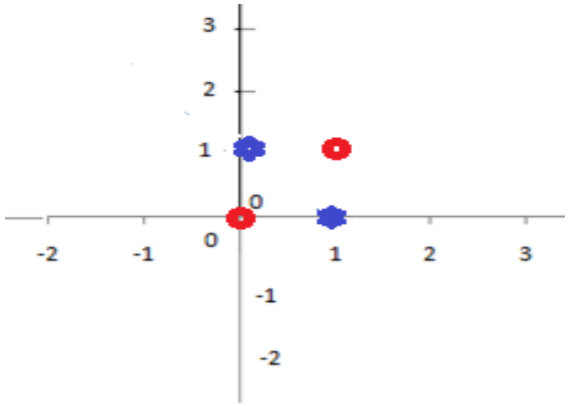


Linear separability

- For 2 dimensional inputs, there exist a line (b1+w1x1+w2x2=0) that separates all samples into one class or another
- A single MP neuron splits the input points into two sets with Points lying on or above the line and points lying below this line
- x2=-b1/w2+(-w1/w2)*x1



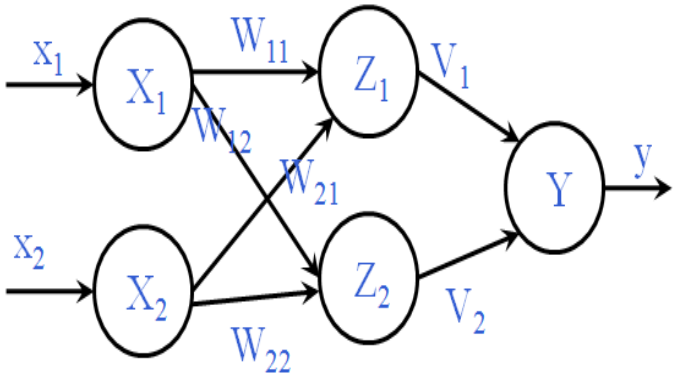
Xor



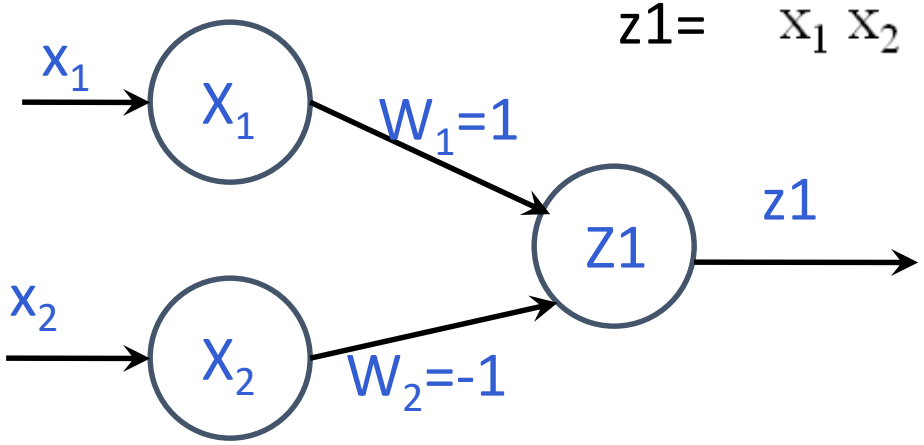
x1	x2	y
0	0	0
0	1	-1
1	0	1
1	1	0

- Not linearly separable
- Can not be represented by single layer network
- Split into components in $x_1 \overline{x_2} + \overline{x_1} x_2$

y=z1+z2 where z1= $x_1 \overline{x_2}$ and z2= $\overline{x_1} x_2$



Implement Z1



z1= $x_1 \overline{x_2}$

x1	x2	z1
0	0	0
0	1	0
1	0	1
1	1	0

If we assume w1=1, w2=-1,

For inputs (0,0) yin= x1w1+x2w2=0*1+0*-1=0

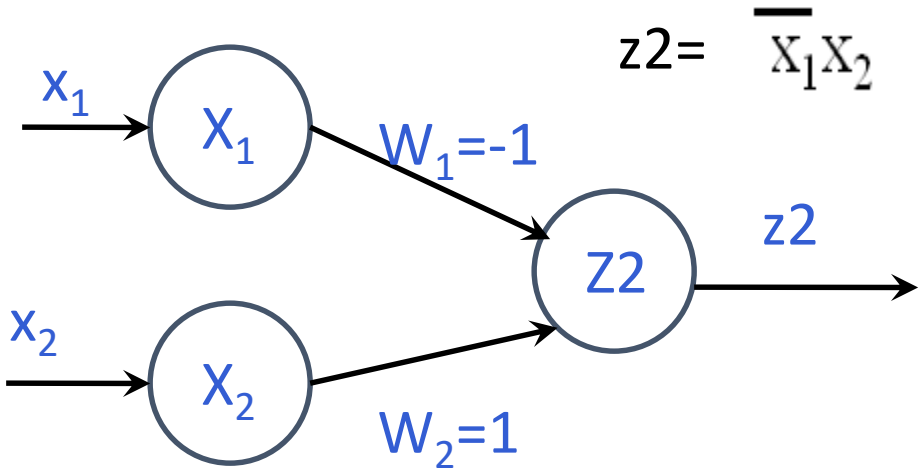
(0,1) yin= x1w1+x2w2=0*1+1*-1=-1

(1,0) yin= x1w1+x2w2=1*1+0*-1=1

(1,1) yin= x1w1+x2w2=1*1+1*-1=0

nw-p $\leq \theta$ nw-p $= 2*1-1 = 1 \leq \theta$ $\theta = 1$

Implement Z2



x1	x2	z2
0	0	0
0	1	1
1	0	0
1	1	0

If we assume w1=-1, w2=1,

For inputs (0,0) yin= x1w1+x2w2=0*-1+0*1=0

(0,1) yin= x1w1+x2w2=0*-1+1*1=1

(1,0) yin= x1w1+x2w2=1*-1+0*1=-1

(1,1) yin= x1w1+x2w2=1*-1+1*1=0

nw-p $\leq \theta$ $\theta = 1$

Implement y=z1+z2

y=z1+z2 where z1= $x_1 \overline{x_2}$ and z2= $\overline{x_1} x_2$

x1	x2	z1	z2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

If we assume v1=1, v2=1,

For inputs (0,0) yin= z1v1+z2v2=0*1+0*1=0

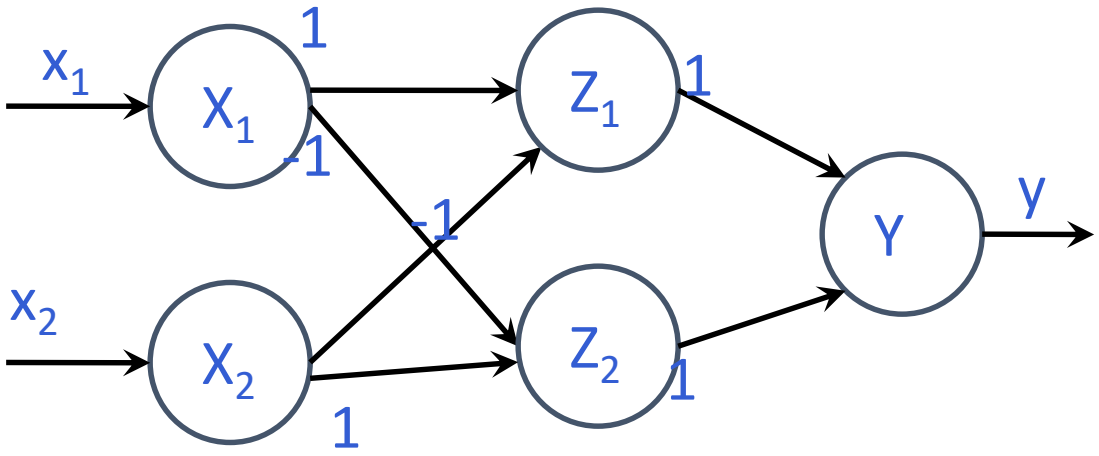
(0,1) yin= z1v1+z2v2=0*1+1*1=1

(1,0) yin= z1v1+z2v2=1*1+0*1=1

(1,1) yin= z1v1+z2v2=0*1+0*1=0

θ can be taken as 1

Final



Assignment(13-2-24)

- Explain Hebb Network (Training algorithm and Flow chart)
- Design a Hebb net to implement OR function (consider bipolar inputs and targets).

Tutorial question(15-2-2024)

Implement NOT, NOR and NAND function using McCulloch-Pitts neuron